KU LEUVEN

FACULTY OF ENGINEERING
DEPARTMENT OF CIVIL ENGINEERING
STRUCTURAL MECHANICS SECTION
KASTEELPARK ARENBERG 40, B-3001 LEUVEN

MACEC 3.3

A MATLAB TOOLBOX FOR EXPERIMENTAL AND OPERATIONAL MODAL ANALYSIS

USER'S MANUAL
JULY 2014
REPORT BWM-2014-06

EDWIN REYNDERS
MATTIAS SCHEVENELS
GUIDO DE ROECK

© 1997–2014 KU Leuven – Structural Mechanics Section
Kasteelpark Arenberg 40, B-3001 Leuven, Belgium.

The software described in this document is distributed under a license agreement. The software may be used only under the terms of the license agreement. No part of this document may be reproduced in any form by print, photoprint, microfilm, or any other means without written permission from the publisher.

Contents

List of abbreviations . . . viii

1 Introduction **1**

1.1 What's new in MACEC 3.3? . . . 1
1.2 What's new in MACEC 3.2? . . . 1
1.3 What's new in MACEC 3.1? . . . 2
1.4 What's new in MACEC 3.0? . . . 3
1.5 A Matlab toolbox . . . 5
1.6 MACEC and SPICE . . . 5
1.7 Development . . . 5

2 Getting started **7**

2.1 MACEC and MATLAB . . . 7
2.2 Operating system and hardware requirements . . . 7
2.3 Installation of MACEC . . . 7
2.4 How to run MACEC . . . 8
2.5 Help and support . . . 8
2.6 Bug reports . . . 9
2.7 Note for SPICE users . . . 9

3 Structure and conventions of MACEC **11**

3.1 Program structure . . . 11
3.2 Variable formats . . . 12
3.3 File structures . . . 13
3.3.1 Measurement data: ascii, mat, ddf, msd, tdm, wav, and f32 formats . . . 13
3.3.2 Sensor definitions: ASCII format . . . 14
3.3.3 Geometry definitions: ASCII format . . . 14
3.3.4 All GUI files: mat format . . . 17
3.4 Graphical User Interface . . . 22

3.5 Logfile and batch run 24

4 Tutorials **27**

4.1 Operational modal analysis of a frame structure 27

4.1.1 Introduction 27

4.1.2 Building the geometry 28

4.1.3 Processing the measured signals 34

4.1.4 Identification of a linear system model 38

4.1.5 Modal analysis of the identified system models 47

4.1.6 Results 51

4.1.7 Conclusions 54

4.2 The B15 bridge 56

4.2.1 Introduction 56

4.2.2 Vibration measurements 57

4.2.3 Building the geometry 59

4.2.4 Processing the measured signals 60

4.2.5 System identification 64

4.2.6 Modal Analysis 67

4.2.7 Conclusions 74

4.3 Roving hammer testing of a rib-stiffened plate 76

4.3.1 Roving hammer testing 76

4.3.2 Rib-stiffened plate 77

4.3.3 Processing the measured signals into an FRF matrix 78

4.3.4 System identification using the pLSCF method 81

4.3.5 Modal analysis after deterministic pLSCF 83

4.3.6 Interpretation and conclusions 83

5 Overview of MACEC functions **85**

5.1 @mcsignal 85

5.1.1 axesACPSDplot 85

5.1.2 axesCPSDFRFCOHplot 87

5.1.3 axesplot 88

5.1.4 bplot 90

5.1.5 concatenate 90

5.1.6 decimate 90

5.1.7 delete 90

5.1.8 detrend . . . 91
5.1.9 df . . . 91
5.1.10 dinkb . . . 91
5.1.11 dinkbf . . . 91
5.1.12 dinkbtau . . . 92
5.1.13 disp . . . 92
5.1.14 display . . . 92
5.1.15 domf . . . 92
5.1.16 dt . . . 93
5.1.17 f . . . 93
5.1.18 f2 . . . 93
5.1.19 f_ . . . 93
5.1.20 f_n_ . . . 94
5.1.21 fdata . . . 94
5.1.22 fdata2 . . . 94
5.1.23 fftfreq . . . 94
5.1.24 filter . . . 95
5.1.25 filtfilt . . . 95
5.1.26 fplot . . . 95
5.1.27 integrate . . . 96
5.1.28 key . . . 96
5.1.29 labels . . . 96
5.1.30 mcsignal . . . 96
5.1.31 md5 . . . 97
5.1.32 n_ . . . 98
5.1.33 nch . . . 98
5.1.34 noelec . . . 98
5.1.35 numel . . . 98
5.1.36 plot . . . 98
5.1.37 quantity . . . 100
5.1.38 resample . . . 100
5.1.39 select . . . 100
5.1.40 setn_ . . . 101
5.1.41 setsifactor . . . 101
5.1.42 sifactor . . . 101

5.1.43 subsref . . . 101
5.1.44 t . . . 102
5.1.45 t_ . . . 102
5.1.46 tdata . . . 102
5.1.47 tplot . . . 102
5.1.48 trim . . . 103
5.1.49 volt2x . . . 103
5.1.50 volt2xa . . . 103
5.1.51 weight . . . 104
5.1.52 window . . . 104
5.2 Conversion . . . 104
5.2.1 cell_ . . . 104
5.2.2 fe2ss . . . 105
5.2.3 input2mcsignal . . . 105
5.2.4 modal2ss . . . 105
5.2.5 rmfd2ss . . . 106
5.2.6 rmfd2sysmat . . . 106
5.2.7 volt2x . . . 106
5.3 Mathematics . . . 107
5.3.1 Kronpr . . . 107
5.3.2 blckComp . . . 107
5.3.3 blckHank . . . 107
5.3.4 blckToep . . . 107
5.3.5 blckVec . . . 108
5.3.6 cov_ . . . 108
5.3.7 eiglr . . . 108
5.3.8 key . . . 108
5.3.9 re_ . . . 109
5.3.10 vec_ . . . 109
5.3.11 vecuns . . . 109
5.4 Modal analysis . . . 109
5.4.1 anpsd . . . 109
5.4.2 cmif . . . 110
5.4.3 drawmodes . . . 111
5.4.4 globmod . . . 112

5.4.5 mac . . . 112
5.4.6 mdtime . . . 113
5.4.7 mod3D . . . 113
5.4.8 modeselect . . . 113
5.4.9 modfind . . . 114
5.4.10 mp . . . 114
5.4.11 mpd . . . 115
5.4.12 mphc . . . 115
5.4.13 mscf . . . 115
5.4.14 npmodselect . . . 116
5.4.15 propmodpar5 . . . 116
5.4.16 ssmodparvar . . . 117
5.4.17 stable_propmodpar5 . . . 118
5.4.18 stabpick . . . 120
5.4.19 stabplot . . . 120
5.5 System identification . . . 121
5.5.1 H1estimate . . . 121
5.5.2 PSDpos_corr . . . 121
5.5.3 PSDpos_per . . . 122
5.5.4 RMFDcalc . . . 122
5.5.5 csi_data . . . 123
5.5.6 identsel . . . 123
5.5.7 makenonpar . . . 124
5.5.8 modalmr . . . 124
5.5.9 pLSCF4 . . . 125
5.5.10 ss2mss . . . 125
5.5.11 ssi_data3 . . . 125
5.5.12 ssicov . . . 126
5.5.13 sysmatcalc . . . 127

6 Verification examples **129**

6.1 Example 1: 2DOF system . . . 129
6.1.1 Example 1a: discretization . . . 130
6.1.2 Example 1b: pLSCF method . . . 131
6.1.3 Example 1c: modal time-domain response decomposition using SSI-data . . . 134

References 137

List of abbreviations

CMIF	complex mode indication function
CSI	combined deterministic-stochastic subspace identification
CSI/ref	reference-based combined deterministic-stochastic subspace identification
DOF	degree of freedom
EMA	experimental modal analysis
FDD	frequency domain decomposition
FRF	frequency response function
GUI	graphical user interface
OMA	operational modal analysis
OMAX	operational modal analysis with exogenous inputs
pLSCF	poly-reference least squares complex frequency domain
PP	peak picking
PSD	power spectral density
PSD+	positive power spectral density
RMFD	right matrix-fraction description
SSI	stochastic subspace identification
SSI-cov	covariance-driven stochastic subspace identification
SSI-cov/ref	reference-based covariance-driven stochastic subspace identification
SSI-data	data-driven stochastic subspace identification
SSI-data/ref	reference-based data-driven stochastic subspace identification
ZOH	zero-order-hold

Chapter 1

Introduction

Welcome to MACEC, a program dedicated to modal analysis

Modal analysis of structures consists of three distinct steps: data collection, system identification and determination of the modal characteristics (eigenfrequencies, damping ratios, mode shapes and modal scaling factors). MACEC is a powerful tool that deals with every step in the modal analysis procedure, except for the data collection. The inputs to the program are raw measurement data. MACEC offers extensive functionalities for the visualization and processing of the data, the identification of system models and the determination and visualization of the structure's modal characteristics. The program disposes of a graphical user interface (GUI), which makes it very intuitive and easy to handle.

1.1 What's new in MACEC 3.3?

This section highlights the changes in MACEC 3.3 with respect to MACEC 3.2. The major changes with respect to MACEC 3.1, 3.0 and 2.0 are listed in sections 1.2, 1.3 and 1.4, respectively.

- The implementation of the covariance-driven stochastic subspace identification (SSI-cov) algorithm has been thoroughly modified, resulting in a **fast and memory efficient computation of the (co)variances** of the estimated system matrices and modal characteristics.
- The possibilities to save animations of mode shapes have been extended, such that it is now possible to **save the animations both on Windows 32-bit and on Windows 64-bit platforms** in `*.avi` format.
- A **new tutorial** has been added, with a double aim: explaining how MACEC can be run in batch mode and how MACEC functions can be integrated into Matlab scripts, and explaining how a roving hammer test can be efficiently processed with MACEC.

1.2 What's new in MACEC 3.2?

This section highlights the changes in MACEC 3.2 with respect to MACEC 3.1. The major changes with respect to MACEC 3.0 and MACEC 2.0 are listed in sections 1.3 and 1.4, respectively.

- Modal parameter estimation from identified nonparametric frequency response functions (FRFs) or positive power spectral densities (PSD^+s) has been made possible by adding new functions

and a new GUI. Although this approach yields usually rather rough modal parameter estimates compared to parametric system identification [8, 10], it is physically very intuitive and therefore mainly suited for getting a first, quick idea of the structural modes before performing a more detailed analysis, or for educational purposes. Two methods are supported:

  - *Peak Picking* (PP), where the modal characteristics are identified by selecting the peaks of the averaged normalized power spectral density function, which is easily computed from an identified PSD^+ function. This technique is only available for output-only data.
  - *Complex Mode Indication Function* (CMIF), where peaks of the singular values of the identified FRF matrix as a function of frequency, are selected for estimating the modal characteristics [16]. When applied to output-only data, the singular values of the PSD matrix, computed as the sum of the identified PSD^+ matrix and its transpose, are used; the method is then sometimes also referred to as *Frequency Domain Decomposition* (FDD). Please note that modal damping ratios are not provided for these methods, as nonparametric damping ratio estimation is not reliable.

- Data import from GeoSIG Miniseed `*.msd` files is now supported. Multiple files, coming from simultaneous measurements on different stations, can be imported at once, and they are automatically synchronized.
- New functions have been added so as to support time-domain modal response decomposition from an identified output-only model. The use of these functions is illustrated in a new verification example (section 6.1.3).
- Minor functionality changes:
  - In the GUI, it is now possible to change the node number and measurement direction attached to the different channels also after system identification and modal analysis. Extra checks have been introduced in order to reduce the risk of human errors when node numbers and measurement directions are defined for different data files at the same time.
  - The computation of modal transfer norms has been reprogrammed, so that the Systems Control Toolbox® of MATLAB is no longer needed for computing them.

1.3 What's new in MACEC 3.1?

- The functionality of the stabilization diagram has been largely extended.
  - If real normal modes are expected, a lower limit value for the modal phase collinearity and an upper limit value for the mean phase deviation can be imposed [5, section A.4.5].
  - A lower bound can be imposed on the mean phase of the mode shapes. This is helpful when real normal modes are expected after mass normalization.
  - If covariance matrices are estimated, which is possible for the SSI-cov/ref subspace identification method, upper limit values for the standard deviations of the eigenfrequencies, damping ratios, and mode shapes can be set.
  - The change in modal transfer norm between two consecutive model orders can be limited.
  - When a mode is pointed to in the stabilization diagram, additional information is shown, such as the modal phase collinearity, the mean phase deviation, the mean phase, and the damping and mode shape standard deviations, when available.

  - When a mode is selected in the stabilization diagram, the mode shape is plotted in the complex plane. For mass-normalized mode shapes, it is possible to plot them as such or using the unit modal displacement weighting scheme.

- Identified and selected modal characteristics are now clustered as fields of a single Matlab STRUCT variable. This choice enhances the flexibility of MACEC when used in batch mode, and it allows for more compact function calls. It affects most MACEC functions in which modal characteristics are involved. These functions have been adapted accordingly.

- When combining modal characteristics from different setups, reference sensors are no longer required if the mode shapes are mass-normalized.

- The function for computing the modal assurance criterion (MAC) values has been completely re-programmed so as to make it much faster. In MACEC 3.0, the square root of the MAC value was computed instead of the MAC value itself; this bug has been removed.

- A new section has been added to the manual, containing verification examples. These examples serve two purposes: (i) demonstrating how the MACEC functions, which are explained in chapter 5, can be employed directly, without the GUI, and (ii) verifying the performance of MACEC by comparing the obtained results with analytical solutions or benchmark results reported in the literature.

- The MACEC function `propmodpar4.m` has been extended so as to allow the computation of the modal characteristics of a *continuous-time* state-space model.

- The format of the sensor definitions files has been changed, and the selection of displacement, strain, and velocity sensors from the file is now supported in the GUI. See section 3.3.2 for more information and an example.

1.4 What's new in MACEC 3.0?

MACEC 3.0 can be used for Experimental, Operational and Combined Modal Analysis of structures, in contrast to previous versions of MACEC, which only dealt with Operational Modal Analysis.

In Experimental modal analysis (EMA), the structure is excited by one or several measured forces, the response of the structure to these forces is recorded and the modal characteristics are extracted from the input-output data. In Operational Modal Analysis (OMA), only the responses are recorded and the modal characteristics are extracted from output-only data. Because the forces are not measured, an extra assumption is needed: in all existing OMA procedures, it is assumed that the unmeasured inputs can be mathematically described as white noise. When a modal test is performed in operational conditions, and when in addition to the operational excitation, one or several measured, artificial forces are applied, it is called an OMAX test (Operational Modal Analysis with eXogenous inputs) [4].

MACEC 3.0 is a *major* upgrade of MACEC 2.0. All sections of the program have been totally re-designed and many new features have been added, such as:

- file management

  - *MACEC 3.0 creates a Matlab file named* `logfile.m` *in the MATLAB working directory, which contains all commands that the user calls using the GUI. This file makes it possible for the user to run MACEC in batch without the GUI*;

  - All files created by the program are in `*.mat` format, which provides a large data reduction and more transparency for the user. Only the gridfiles, slavefiles and beam/surface files are still in ASCII format, as the information needed to create these files is often available as ASCII data;
  - MACEC 3.0 makes it much easier to load multiple files;
  - In MACEC 3.0, it is possible to import data from NI LabVIEW `*.tdm` files and LMS Test.Lab `*.wav` files. Formats that were supported in previous MACEC versions can still be imported.

- geometry
  - the definition of measurement nodes and connections between these nodes is now graphically supported in the GUI so that you can detect related errors in an early stage;
  - surfaces are not longer defined as quadrilaterals but as triangles, so that a defined surface is always a plane surface.

- signal processing
  - MACEC 3.0 makes use of Matlab's Signal Processing Toolbox and of the Matlab toolbox Sigfun that has been developed at the Structural Mechanics Section of KU Leuven [15];
  - the measurement data that serve as input for MACEC are converted to an mcsignal object, on which all signal processing operations are performed.

- system identification
  - for stochastic subspace identification, not only the data-driven stochastic subspace identification (SSI-data) method and its reference-based generalization (SSI-data/ref) are available, but also the covariance driven versions SSI-cov and SSI-cov/ref [7];
  - with SSI-cov(/ref), it is possible to calculate covariances on the system estimates [14];
  - combined subspace identification is now possible, with the combined deterministic-stochastic subspace identification (CSI) and the reference-based combined deterministic-stochastic subspace identification (CSI-ref) methods [12];
  - the poly-reference least squares complex frequency domain (pLSCF) algorithm, also known under its commercial name Polymax[1], which has become a standard for both experimental and operational modal testing, has been added. Deterministic and stochastic pLSCF are available;
  - the Peak Picking method is no longer supported as simulation studies have shown that it is clearly inferior to the identification methods that are available in the toolbox [8].

- modal analysis
  - the stabilization diagram has been extended with a lot of functionalities, such as: much more stabilization criteria (damping range, maximum frequency variance, modal transfer norm) are available, the mode shapes are plotted in the complex plane as soon as a pole is picked, the stabilization diagram can be saved as a Matlab `*.fig` file, ...;
  - the modes are automatically scaled to unity modal mass if at least one force and one output are measured at the same DOF, otherwhise they are scaled to unit modal displacement;
  - the animation of mode shapes with surfaces is improved;
  - mode shape animations can be saved as `*.avi` files.

[1] LMS PolyMAX is a registered trademark of LMS International

1.5 A Matlab toolbox

MACEC is not a stand-alone program, but a toolbox for use with MATLAB® and the Signal Processing Toolbox of MATLAB.

1.6 MACEC and SPICE

MACEC is a part of the SPICE program, but it also runs separately. SPICE coordinates the MACEC and SASW programs, which both make use of the same preprocessing features.

1.7 Development

MACEC 1.0 started in 1997-1998 as the Master Thesis project of Bart Van den Branden and Alexander Laquière at the Civil Engineering Department of KU Leuven under the supervision of Bart Peeters and prof. Guido De Roeck. Between 1998 and 2001, the program was improved and extended into version 2.0 by Bart Peeters in the framework his Ph.D Thesis [6]. Between 2006 and 2011, the program was totally re-designed and many new features were added by Edwin Reynders in the framework of his Ph.D. Thesis [10] and subsequent postdoctoral research, resulting in the 3.0, 3.1, 3.2 an 3.3 versions. The mcsignal class was originally created by Mattias Schevenels as a part of of the `sigfun` toolbox [15]. The class was later incorporated into the `MACEC` toolbox instead of the `sigfun` toolbox.

Edwin Reynders
Leuven, July 9, 2014.

Chapter 2

Getting started

2.1 MACEC and MATLAB

Before you can run MACEC on your computer, you need to have a compatible MATLAB version installed. MACEC 3.3 has been tested on MATLAB 8.2.0 (R2013b).

Important: In contrast to MACEC 2.0 *and older, the 3.3 version requires that you have the* Signal Processing Toolbox® *of* MATLAB *installed.*

2.2 Operating system and hardware requirements

MACEC can be used on the Windows® operating system only. Both the 32-bit and the 64-bit versions are supported.

2.3 Installation of MACEC

The installation of MACEC consists of the following steps:

1. Install MATLAB on your computer;
2. Copy the `spice33-2014-07a` and `sigfun-2008-10a` folders to your hard disc, for instance to the directory

   `C:\Program Files\MATLAB\R2013b\toolbox`

3. Move your license file, named `splicense.mat`, to the `license` subdirectory of `spice33-2014-07a`;
4. Start MATLAB;
5. In the main window of MATLAB, choose **FILE** → **SET PATH**. Then click ADD WITH SUBFOLDERS and specify the `spice33-2014-07a` and `sigfun-2008-10a` directories or simply

   `C:\Program Files\MATLAB\R2013b\toolbox`

   Then click SAVE and then CLOSE.

2.4 How to run MACEC

After the installation of MACEC has been completed, first specify the directory you want to work in by changing the CURRENT DIRECTORY field in the MATLAB main window. Now you can start the GUI of MACEC simply by typing

macec

in the Command Window of Matlab and then press ENTER. MACEC starts up and the main window of MACEC appears:

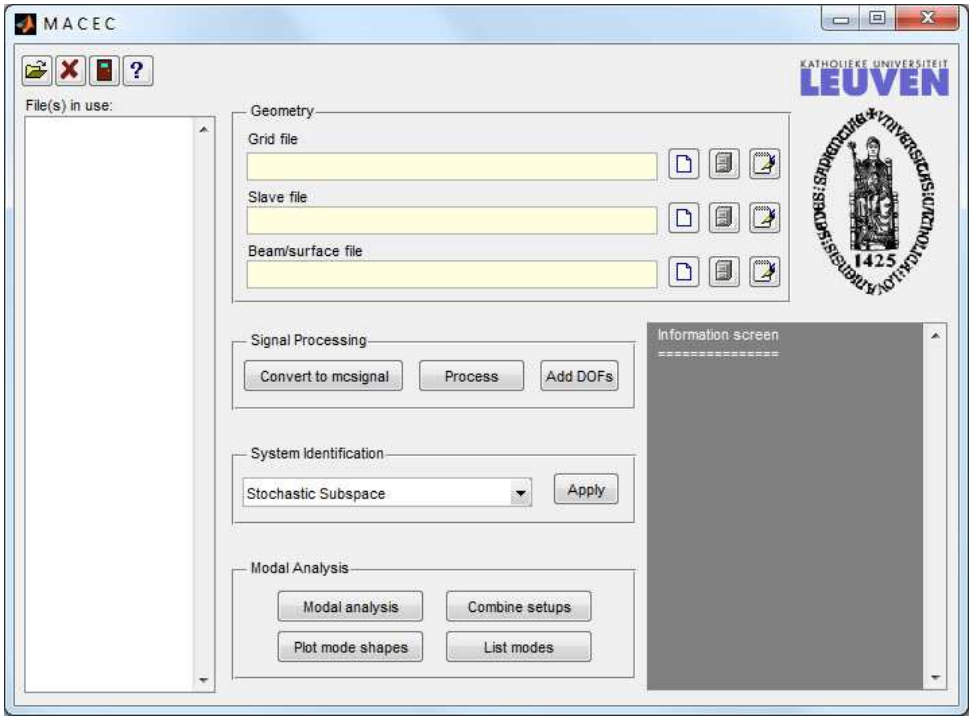


Figure 2.1: The main window of MACEC

You can also call individual MACEC functions (see chapter 4) from the Matlab Command Window or by running a Matlab `*.m` file.

2.5 Help and support

This manual provides the main user support for MACEC. It also contains tutorials which clarify the use of the MACEC GUI. Furthermore, detailed help for the individual MATLAB functions of MACEC is available from this manual (chapter 5), as well as examples of the use of these functions in batch mode (chapter 6).

Important: Please note that for MACEC, no help desk or individual support is available. The manual and the MATLAB help should provide sufficient user support.

2.6 Bug reports

MACEC has been extensively tested and validated. However, it might be possible that you still encounter bugs. If so, please report them by email to `macec` (at) `bwk` (dot) `kuleuven` (dot) `be`.

2.7 Note for SPICE users

MACEC s part of the SPICE program, which contains besides MACEC also the SASW module for Spectral Analysis of Surface Waves[1]. You can start the SPICE program simply by typing

spice

in the Command Window of Matlab and then press Enter. SPICE first starts the MACEC module. If you have a license for SASW, you can easily switch to the SASW module by making use of the SASW button at the top of the SPICE main window.

[1] In SPICE 3.0 and newer, the tools for vibration analysis in the build environment are no longer available. They are now part of Sigfun, the signal Processing toolbox of the Structural Mechanics Section of KU Leuven. When purchasing MACEC, you also obtain Sigfun. Please consult the Sigfun user's guide [15] for more information.

Chapter 3

Structure and conventions of MACEC

By reading this chapter, you will become familiar with the structure of the MACEC program and its main conventions regarding variable formats, file formats and visualization, and you will learn how to use the logfile.

3.1 Program structure

MACEC is a part of the SPICE MATLAB toolbox, and it makes use of another MATLAB toolbox developed at the Structural Mechanics Section of KU Leuven: SIGFUN. The MATLAB toolboxes are a set of MATLAB functions that serve a specific purpose:

- SPICE contains functions that allow the user to identify a mathematical system model from measured data (system identification), to perform a modal analysis with the identified system description and to perform a spectral analysis of surface waves from measured data;
- SIGFUN contains functions that are useful for the processing of measured signals.

If you take a look at the functions that are available in the SPICE toolbox, you notice that they are divided into several maps:

- `GUI` contains all functions that are needed to run the SPICE/MACEC graphical user interface;
- `conversion` contains all functions that deal with the conversions between data types, for instance the conversion from measured data in ASCII format to a MATLAB mcsignal object;
- `mathematics` contains all mathematical functions that are useful in system identification but not available in MATLAB, for instance for constructing a block Hankel matrix;
- `modal analysis` contains all functions that can be used for the modal analysis of an identified system model, for instance for determining the modal characteristics of an identified Right Matrix Fraction Description model;
- `sasw` contains all functions related to spectral analysis of surface waves;
- `system identification` contains functions that can be used for system identification, for instance Reference-based Combined Deterministic-Stochastic Subspace Identification (CSI/ref);
- `@dummy` contains the functions for objects of the DUMMY type;

- `@mcsignal` contains the functions for objects of the MCSIGNAL type.

For a good understanding of terminology it is important to make the distinction between system identification and modal analysis. Experimental, operational or combined modal analysis as defined in the literature consists of three distinct steps:

1. collection of the data and preprocessing
2. system identification
3. determination of the modal characteristics from the identified system model (modal analysis)

So the term "modal analysis" is defined at two different levels: (i) the whole process of obtaining modal characteristics from measurements (steps 1, 2 and 3) and (ii) the determination of the modal characteristics from the identified system model (step 3). This often causes confusion, which sometimes results in mixing up system identification with modal analysis of level ii, especially when the difference is subtle, for instance when modal characteristics are determined directly from a nonparametric frequency response function (FRF), identified from data. Therefore, the MACEC toolbox clearly differentiates between system identification and modal analysis of level ii, both in the function definitions and in the GUI.

3.2 Variable formats

MACEC makes use of standard MATLAB variable formats as DOUBLE, CHAR, STRUCT, etc. Consult the MATLAB help for more information.

In addition, MACEC also allows the user to create MCSIGNAL (multi-channel signal) variables, which are, from a programming point of view, objects of the MCSIGNAL class. The philosophy behind MCSIGNAL objects is that they contain all information about the measured data in one single MATLAB variable. Therefore, MCSIGNAL objects have the following properties:

- NCH is the number of channels used in the measurement;
- N is the number of samples measured;
- F is the *sampling* frequency (in Hz);
- DF is the frequency resolution (in Hz);
- T is the total measurement duration (in s);
- DT is the sampling period (time resolution) (in s);
- QUANTITY is a cell containing the measured quantities of each channel (for instance `acc` stands for acceleration);
- SIFACTOR contains the factor which links the measurement units to SI units (for instance if the measured data for a particular channel are accelerations and the sifactor equals 10^{-3}, the units are mm/s^2);
- LABELS is a cell containing the labels of each channel.

The properties of an MCSIGNAL object are accessible in two ways: in dot notation or as a command. For instance, if you want to know the sampling frequency of an MCSIGNAL object named CNVDAT, you can use the MATLAB commands[1]

cnvdat.F or *f_(cnvdat)*

Objects of the mcsignal type are defined with the *mcsignal* command. For more information about this command and for an overview of all functions that can be performed on objects of the MCSIGNAL type, please consult section 5.1.

3.3 File structures

3.3.1 Measurement data: ascii, mat, ddf, msd, tdm, wav, and f32 formats

MACEC contains several functions to load measurement data from external files into MATLAB and to convert these data to an mcsignal object (see section 3.2):

- Measurement data that were stored, during the data acquisition, using DASYLAB® software, are available in `*.ddf` format. They can be converted to an MCSIGNAL object with the GUI, the *input2mcsignal* command (see section 5.2.3) or the *mcsignal* command (see section 5.1);
- Measurement data that were stored, during the data acquisition, in GeoSIG® measurement stations, are available in `*.msd` (Miniseed) format. They can be converted to an MCSIGNAL object with the GUI, the *input2mcsignal* command (see section 5.2.3) or SIGFUN's *readmsd* and MACEC's *mcsignal* commands (see section 5.1);
- Measurement data that were stored, during the data acquisition, using LabVIEW® software, are available in `*.tdm` format. They can be converted to an MCSIGNAL object with the GUI, the *input2mcsignal* command (see section 5.2.3) or the *mcsignal* command (see section 5.1);
- Measurement data that were stored, during the data acquisition, using Test.Lab® software, are available in `*.wav` format. They can be converted to an MCSIGNAL object with the GUI, the *input2mcsignal* command (see section 5.2.3) or the *mcsignal* command (see section 5.1);
- Measurement data that are available in `*.asc` format (ASCII type), can be converted to an MCSIGNAL object with the GUI, the *input2mcsignal* command (see section 5.2.3) or the *mcsignal* command (see section 5.1);
- Measurement data that are available in `*.mat` (MATLAB binary) format, can be converted to an MCSIGNAL object with the *mcsignal* command (see section 5.1);
- Measurement data that are available in `*.f32` format, can be converted to an MCSIGNAL object with the GUI or the *input2mcsignal* command (see section 5.2.3); The `*.f32` format has been used at the Civil Engineering Department of KU Leuven to convert measurement data that were available in ASCII format to a binary format in order to save disk space.

[1] Please note that these commands are case sensitive.

3.3.2 Sensor definitions: ASCII format

If you use the CONVERSION OF THE MEASURED DATA window of the MACEC GUI, it is possible to load a file containing sensor definitions for the conversion of the measurement data. This file should be of ASCII format. Five types of sensors can be defined: accelerometers, force sensors, displacement sensors, strain sensors, and velocity sensors.

Each sensor is defined on a single row. A sensor definition consists of 5 or more columns which contain, from left to right:

- the sensor type;
- the sensor number;
- the manufacturer type;
- the serial number;
- the sensitivity;
- extra columns may be added, but they are not used by MACEC.

The sensor type is a string: `A` denotes an accelerometer with raw measurements in Volt, `AC` an accelerometer with raw measurements in digital counts, `F` a force sensor, `D` a displacement sensor, `S` a strain sensor, and `V` a velocity sensor.

For accelerometers, the sensitivity is defined in [mV/g] or [counts/g]; for force sensors, in [mV/lbf]; for displacement sensors, in [V/m]; for strain sensors, in [mV/e]; and for velocity sensors, in [V/(m/s)].

Comments may be used, but need to be put between two `#` symbols. An example of a sensor definitions file is shown in fig. 3.1.

3.3.3 Geometry definitions: ASCII format

All files that deal with the geometry of the measurement setup, need to be of the ASCII type. See the subsections below for the definitions of specific geometry-related files.

Grid files

In MACEC, the files that contain the definition of the measurement nodes are called grid files. They can have any name, as long as they are of the ASCII type (extension `*.asc`). A grid file is structured as follows:

- each row contains the definition of a measurement node;
- each row has four columns, containing the node number[2] and its (X, Y, Z) coordinates, in any units.

Example. Figure 3.2 shows a small grid file, containing the definitions of 4 nodes. Node 1 has coordinates $(0, 0, 0)$, node 2 is situated at $(3, 0, 0)$, node 3 at $(0, 3, 0)$ and node 4 at $(3, 3, 0)$.

[2] Only positive nonzero integers are allowed as node numbers.

```
#ACCELEROMETERS#

#type no  manufacturer   SN      sens(mV/g)#
#---------------------------------------#

 A    31  PCB_393A03     3229    982
 A    32  PCB_393A03     3230    977
 A    33  PCB_393A03     3231    982
 A    34  PCB_393A03     3232    996

 A    91  PCB_393B04     11042    995
 A    92  PCB_393B04     11043    987
 A    93  PCB_393B04     11045    998
 A    94  PCB_393B04     11046    997

#type no  manufacturer   SN      sens(counts/g)#
#-----------------------------------------#

 AC 1931  GS-GMS18(trix) 1931    3774874
 AC 1932  GS_GMS18(triy) 1932    3774874
 AC 1933  GS_GMS18(triz) 1933    3774874

FORCE_SENSORS#

#type no   manufacturer  SN     sens(mV/lbf)#
#---------------------------------------#

 F    111  PCB_208A05    11569  0.986
 F    114  PCB_208C05    20170  1.041
 F    115  PCB_208C05    25930  1.006

DISPLACEMENT_SENSORS

#type no   manufacturer        SN    sens(V/m) range(mm)#
#-----------------------------------------------------#

 D    201  micro-eps_S601-0.2  1248  50000     0.2
 D    202  micro-eps_S601-0.5  1164  50000     0.5
 D    203  micro-eps_S601-01   1258  50000     2.0

STRAIN_SENSORS

#type no   manufacturer   SN    sens(mV/e)  freq_range(Hz)  ampl_range(pk_e)  res(e)#
#-------------------------------------------------------------------------------#

 S    301  PCB_740B02     3937  53.7         0.5-100000      100               0.6
 S    302  PCB_740B02     3938  53.6         0.5-100000      100               0.6
 S    303  PCB_740B02     3939  50.8         0.5-100000      100               0.6
 S    304  PCB_740B02     3940  45.7         0.5-100000      100               0.6
  VELOCITY_SENSORS

#type  no  manufacturer  SN  sens(V/(m/s))#
#--------------------------------------#

#no velocity sensors have been defined yet#
```

Figure 3.1: Example of a sensor definitions file

1	0	0	0
2	3	0	0
3	0	3	0
4	3	3	0

Figure 3.2: Example of a grid file

Slave files

Slaving is a powerful procedure for the visualization of DOFs that are not measured, but that can assumed to be related to measured DOFs in a linear way. A specific DOF can be slaved to several master DOFs; the results are then additive.

In MACEC, the files that contain the definition of the dependent degrees of freedom (DOFs) are called slave files. They can have any name, as long as they are of the ASCII type (extension `*.asc`). A slave file is structured as follows:

- in each row, one DOF is coupled to one other DOF;
- each row has eight columns, containing the node number of the master node and the master DOF (direction in (X, Y, Z) coordinates), followed by the node number of the slave node and the slave DOF (amplified direction in (X, Y, Z) coordinates), respectively.

Example. Figure 3.3 shows a small slave file, containing the definitions of 4 slave DOFs:

- In the first row, the deformation in the X direction of node 7 is coupled to the deformation in the X direction of node 2. The amplification factor is 1;
- In the second row, the deformation in the X direction of node 8 is coupled to the deformation in the X direction of node 3. The amplification factor is 0.5, so that the amplitude of the $8X$ DOF is half that of the $3X$ DOF;
- In the third row, the deformation in the X direction of node 8 is coupled to the deformation in the X direction of node 4. The amplification factor is 0.5, so that the amplitude of the $8X$ DOF is half that of the $4X$ DOF. Because the $8X$ DOF was also coupled to the $3X$ DOF, the amplitude of the $8X$ DOF is the mean of $3X$ and $4X$;
- In the fourth row, the deformation in the X direction of node 6 is coupled to the deformation in the X direction of node 2. The amplification factor is 0, so that node 6 does not move. This "trick" can in general be used for the visualization of DOFs that are not measured but that can, from the geometry of the problem, be supposed to have (nearly) zero amplitude.

Beam/surface files

In MACEC, the files that contain the definition of the connections between the measurement nodes are called beam or surface files, depending on the type of connection you want for the visualization[3].

[3]Since beams and surfaces are only used for the visualization, they are not numbered.

2	1	0	0	7	1	0	0
3	1	0	0	8	0.5	0	0
4	1	0	0	8	0.5	0	0
2	1	0	0	6	0	0	0

Figure 3.3: Example of a slave file

If you want the nodes to be connected by lines, you need a beam file. If you want the nodes to be connected by triangles, you need a surface file. Beam or surface files can have any name, and are of the ASCII type (extension `*.asc`). A beam or surface file is structured as follows:

- each row contains the definition of a beam connection or a surface connection;
- each row contains the numbers of the two nodes that are connected by a beam, or the numbers of the three nodes that are connected by a triangle.

Beams and surfaces can also be combined. In this case, beams should be treated as surfaces for which the last node number equals zero.

Example. Figure 3.4 shows a small beam file. The nodes 1 and 2, 2 and 3, 3 and 4 and 4 and 5 are connected by a beam.

1	2
2	3
3	4
4	5

Figure 3.4: Example of a beam file

3.3.4 All GUI files: mat format

If an experimental, operational or combined modal analysis is performed with the MACEC GUI, you need to save the data, after some operations, in a `*.mat` file. If you have saved the data, the name of the new `*.mat` file appears in the FILE(S) IN USE section of the MACEC main window (figure 2.1). You can proceed further by clicking on the name of the file and by subsequently pressing a command button in the MACEC main window. Please note that the `*.mat` file is *not* loaded in the memory *until* you press a command button.

A powerful possibility of MACEC is that you can also create these `*.mat` files without the GUI, and then use the GUI in a next step by adding the file to the FILE(S) IN USE section of the GUI main window with the SELECT NEW DATA... button (figure 4.7). However, in this case, it is important that the variables stored in the `*.mat` files have the right names, types and dimensions.

Converted measurement data

After you have converted the measurement data (in `*.ddf`, `*.asc`, `*.tdm`, `*.wav`, `*.mat` or `*.f32` format) to an MCSIGNAL object using the GUI, you are required to save this MCSIGNAL object in a `*.mat` file. This `*.mat` file contains only one variable, namely the MCSIGNAL object, which has the name CNVDAT (from "converted data").

Conversion factors

In the CONVERSION OF THE MEASURED DATA window of the MACEC GUI, it is possible to save the conversion factors separately into a `*.mat` file and to load this file later on. This is particularly handy if the measurements are performed in several setups for which the number of channels and the sensors are identical. The saved file contains the following variables:

- LABELS, a cell which contains the labels given to each channel;
- TYPE, a vector containing the information about the data type. If an element equals 1, the corresponding data type is acceleration, if 2 a displacement, if 3 a force, if 4 a strain and if 5 a velocity;
- NUMBER, a vector containing in each element the row number of the sensor definition file (see section 3.3.2) that is associated with a channel;
- SENSITIVITIES, a vector containing the conversion factor of each channel's sensor;
- AMPLIFICATIONS, a vector containing the amplification factors for each channel;
- DB, a vector of which an element equals 1 if the corresponding conversion factor is in deciBel (dB), otherwise it is 0;
- FACTORS, a vector containing the information about the measurement units. If the data is measured in kiloVolt or 10^3 digital counts, the corresponding element in factors equals 1, if it is Volt or digital counts, it equals 2, if it is miliVolt, it equals 3 and if it is microVolt, it equals 4.

Processed measurement data

After you have processed an MCSIGNAL object using the GUI, you need to save this adapted MCSIGNAL object in a `*.mat` file. This `*.mat` file contains only one variable, namely the processed MCSIGNAL object, which has the name PREDAT (from "preprocessed data").

After processing, you can specify the node numbers and measurement directions that are associated with a certain channel. If you use the GUI for this, you are asked if the specifications should be saved or not. If you choose to save them, they are added to the `*.mat` file that already contains the preprocessed data. This file now contains 3 variables:

- PREDAT, the mcsignal object with the preprocessed measurement data;
- NODE_NUM, a vector containing the node numbers that correspond to the channels;
- MEAS_DIR, a matrix containing the angles α and β that correspond to the channels (figure 3.5). Each row corresponds to a channel, the first column contains the α values and the second column the β values.

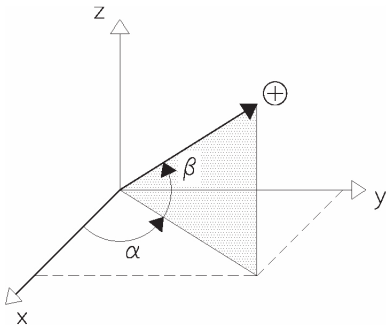


Figure 3.5: Definition of the direction angles α and β.

Measurement DOFs

In the Add channel specifications window of the MACEC GUI, it is possible to save the measurement nodes and measurement directions into a `*.mat` file and to load this file later on. This is particularly handy if in the same measurement setup, different measurements are performed. The saved file contains the following variables:

- nodenumbers is a vector containing the node numbers attached to each channel;
- measurement_directions is a matrix containing in each row, the α and β angles (in the first and the second column, respectively) which determine the measurement direction (figure 3.5).

System Identification results

After you have performed a system identification using the GUI, you have to save the identified system information in a `*.mat` file.

State-space models. If the identified system models are state-space models (identified with SSI-data(/ref), SSI-cov(/ref) or CSI(/ref)), this `*.mat` file contains the following variables:

- predat, see the previous paragraph;
- sysmat, a MATLAB variable of the struct type with the following fields:
  - The system matrices for the different orders. For instance, for a stochastic state-space model of order two that has been estimated with the SSI-data method, the following matrices are present: sysmat.A_2, sysmat.C_2, sysmat.Q_2, sysmat.R_2, sysmat.S_2;
  - sysmat.orders, a vector containing the orders of the systems that have been identified;
  - sysmat.chanselout, a vector containing the channel numbers which have been used as outputs in the identification;
  - sysmat.chanselin, a vector containing the channel numbers which have been used as inputs in the identification (only differs from the empty vector if CSI(/ref) identification has been used).
- node_num, see the previous paragraph;
- meas_dir, see the previous paragraph;

- QUANTS, a cell containing the data type of each channel (for instance ACC stands for acceleration);
- CHAN_OUTIN, a vector containing, for each input, the output number (*NOT* the channel number) that has the same DOF. If no such output is available, the corresponding element of CHAN_OUTIN equals 0;
- AMPL_OUTIN, a vector containing, for each element of CHAN_OUTIN, the corresponding sign. If for example the input at a certain node was measured in the $-X$ direction and the output in the X direction, the corresponding element of CHAN_OUTIN equals -1.

RMFD models. If the identified system models are right matrix-fraction description (RMFD) models (identified with Deterministic pLSCF, Operational pLSCF or Combined pLSCF), the `*.mat` file contains the following variables:

- PREDAT;
- RMFD, a MATLAB variable of the STRUCT type with the following fields:
  - The system matrices for the different orders. For instance, for an RMFD model of order two, the following matrices are present: RMFD.A_2 and RMFD.B_2;
  - RMFD.ORDERS, a vector containing the orders of the systems that have been identified;
  - RMFD.CHANSELOUT, a vector containing the channel numbers which have been used as outputs in the identification;
  - RMFD.CHANSELIN, a vector containing the channel numbers which have been used as inputs in the identification (equals the empty vector if Operational pLSCF identification has been used);
  - RMFD.REFS, a vector containing the channel numbers which have been used as reference outputs in the identification (equals the empty vector if Deterministic pLSCF identification has been used);
  - RMFD.H_MEAS, a 3D matrix containing the estimated FRF's or PSD$^+$'s that have been used as inputs for the pLSCF algorithm The rows correspond to the output channels, the columns to the input channels or the reference output channels and the depth to the frequency points.
- NODE_NUM;
- MEAS_DIR;
- QUANTS;
- CHAN_OUTIN;
- AMPL_OUTIN;
- REFS, a vector containing the output numbers (*NOT* the output channel numbers, they are in RMFD.REFS) which have been used as reference outputs in the identification.

Nonparametric model. If the identified system model is a nonparametric frequency response function (FRF) or positive power spectral density (PSD^+), the `*.mat` file contains the following variables:

- NONPAR, a MATLAB variable of the STRUCT type with the following fields:
  - NONPAR.FREQSCALE, a vector containing the frequency points of the nonparametric model;
  - NONPAR.FRF, a 3D matrix containing the estimated FRF's. The rows correspond to the output channels, the columns to the input channels and the depth to the frequency points. This field can be empty;
  - NONPAR.PSDP, a 3D matrix containing the estimated PSD^+'s. The rows correspond to the output channels, the columns to the reference output channels and the depth to the frequency points. This field can be empty;
  - NONPAR.CHANSELOUT, a vector containing the channel numbers which have been used as outputs in the identification;
  - NONPAR.CHANSELIN, a vector containing the channel numbers which have been used as inputs in the identification. This field can be empty;
  - NONPAR.REFS, a vector containing the channel numbers which have been used as reference outputs in the identification. This field can be empty;
- NODE_NUM;
- MEAS_DIR;
- QUANTS;
- CHAN_OUTIN;
- AMPL_OUTIN.

Modal Analysis results

After you have selected the proper modal characteristics in the stabilization diagram of the GUI (after parametric system identification), or from a nonparametric ANPSD/CMIF/FDD plot (after nonparametric system identification), you are required to save them in a `*.mat` file. This file contains the following variables:

- STABMODES, a MATLAB variable of the STRUCT type with the following fields:
  - STABMODES.F, a vector containing the eigenfrequencies;
  - STABMODES.O, a vector containing the model orders;
  - STABMODES.XI, a vector containing the damping ratios;
  - STABMODES.M, a matrix containing the mode shapes in each column;
  - STABMODES.MTN, a vector containing the modal transfer norms;
  - STABMODES.MP, a vector containing the mean phases;
  - STABMODES.MPC, a vector containing the modal phase collinearities;
  - STABMODES.MPD, a vector containing the mean phase deviations;
  - STABMODES.WSCHEME, a string that indicates whether the mode shapes are mass normalized (`mass`) or scaled to unit modal displacement (`unit`);

  - STABMODES.QUANTS, a cell containing the physical quantities corresponding to each mode shape component;
  - STABMODES.STDF, a vector containing the standard deviations of the eigenfrequencies;
  - STABMODES.STDXI, a vector containing the standard deviations of the damping ratios;
  - STABMODES.STDMR, a matrix containing the standard deviations of the real parts of the mode shapes in each column;
  - STABMODES.STDMI, a matrix containing the standard deviations of the imaginary parts of the mode shapes in each column;
  - STABMODES.STDMMAX, a vector containing the maximum standard deviations of a real or imaginary part of the mode shapes. Please note that, depending on the system identification method that has been employed, some of these fields may not be available.

- NODE_NUM, see the previous paragraph;
- MEAS_DIR, see the previous paragraph;

3.4 Graphical User Interface

MACEC's Graphical User Interface (GUI) is constructed around one main window, divided into different sections. In each section, a specific part of the modal analysis process is dealt with, mostly by clicking buttons which call other windows. In this section, you can find a schematic overview of how the different GUI windows are called. Fig. 3.6 provides an overview of the GUI structure of MACEC and also indicates the interaction between the GUI figures and the files that are created from or interact with the GUI.

The MACEC main window (figure 2.1), which is constructed by calling the *GUI_main* function, consists of the following sections:

- FILE(S) IN USE: In this section, no other MACEC windows are called. It just consists of a list of files that are used in MACEC and buttons to select new files, to quit the main window, etc.;
- GEOMETRY: This section deals with the definition and the editing of grid, slave or beam/surface files. Depending on the button used, the following MACEC functions will be called:
  - *GUI_grid* constructs the window for constructing or editing the grid file;
  - *GUI_slave* constructs the window for constructing or editing the slave file;
  - *GUI_beam* constructs the window for constructing or editing the beam or surface file.
- SIGNAL PROCESSING: This section deals with the construction of an mcsignal object:
  - With the CONVERT TO MCSIGNAL button, the GUI_CONVERT function is called which constructs the window for the conversion of the measurement data to the CNVDAT MCSIGNAL object;
  - With the PROCESS button, the GUI_PREP function is called which constructs the window for signal processing:
    * If the standard MATLAB function *filtfilt* needs to be called during signal processing, the signal processing window calls the *GUI_filtfilt* function, which constructs a dialog window for the FILTFILT operation;

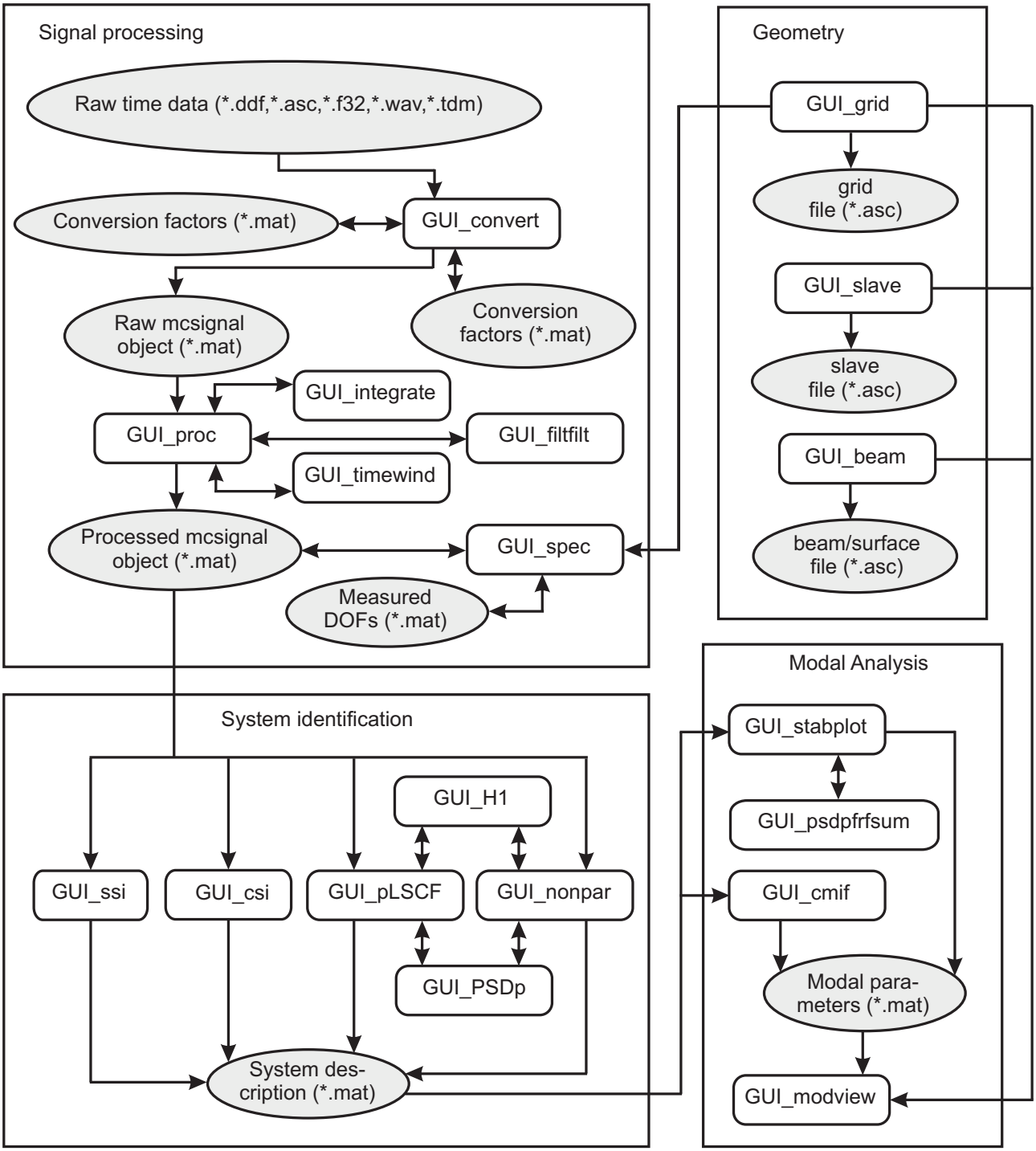


Figure 3.6: GUI structure of MACEC. The GUI figures are indicated by white rectangles, the files that are created from or interact with a GUI figure are indicated by grey ellipses.

    - If a time window needs to be applied during signal processing, the *GUI_timewind* function, which constructs a dialog window, is called by the signal processing window.
  - With the ADD DOFS button, the *GUI_spec* function is called, which constructs the window for the specification of the measurement DOF's.

- SYSTEM IDENTIFICATION: In this section, the user specifies which system identification method he or she wishes to use. Depending on the method chosen, different windows are opened:
  - For all stochastic subspace identification methods (SSI-data, SSI-data/ref, SSI-cov, SSI-cov/ref), the *GUI_ssi* function is called, which constructs the window for SSI;
  - For all combined subspace identification methods (CSI and CSI/ref), the *GUI_csi* function is called, which constructs the window for CSI;
  - For all pLSCF methods (Deterministic pLSCF and Operational pLSCF), the *GUI_pLSCF* function is called, which constructs the window for pLSCF identification. Because an identified non-parametric system model is needed for pLSCF, from this window one of the following functions is called:
    - *GUI_H1* constructs the window for the estimation of a non-parametric FRF (H_1 estimator);
    - *GUI_PSDp* constructs the window for the estimation of a non-parametric Positive Power Spectral Density (PSD+).
  - For nonparametric identification, the *GUI_nonpar* function is called, which constructs the window for nonparametric system identification. From this window, one of the following functions are called:
    - *GUI_H1* for FRF estimation;
    - *GUI_PSDp* for PSD$^+$ estimation.

- MODAL ANALYSIS: This section deals with the modal analysis of the identified system models:
  - By clicking the MODAL ANALYSIS button, one of the following functions is called:
    - If parametric system descriptions have been identified, the *GUI_stabplot* function is called which constructs the window with the stabilization diagram:
      - In this window, the function *GUI_psdpfrfsum* may be called for the construction of a dialog box for the calculation of the sum of the FRFs or PSD+s.
    - If nonparametric system descriptions have been identified, the *GUI_cmif* function is called which constructs the window for modal parameter selection through the peak picking or CMIF/FDD methods.
  - By clicking the PLOT MODE SHAPES button, the *GUI_modview* function is called which constructs the window for the visualization of the mode shapes.

3.5 Logfile and batch run

MACEC has the flexibility that its functions can not only be called from the GUI, but also from the MATLAB command window or from a MATLAB command (`*.m`) file. See chapter 5 for more information about the functions of MACEC that can be called in MATLAB, and chapter 6 for examples where MACEC is used in batch mode.

As soon as you start the MACEC GUI, a MATLAB command file named `logfile.m` is created in the MATLAB working directory. When you perform certain operations in the GUI, this logfile is automatically filled with the proper commands. Such a logfile has three advantages:

1. It leaves you a blueprint of the operations performed in the GUI, which enables you to detect possible errors;
2. It enables you to recalculate everything without the need to perform all interactive operations again;
3. It gives you an example of how to perform a modal analysis by MACEC without the GUI.

Example. Figure 3.7 shows the logfile used for the determination of the modal characteristics in tutorial 1 with the SSI-data/ref identification algorithm. The logfile was automatically created by MACEC.

```
% MACEC version 3.3
% Copyright KU Leuven, Structural Mechanics Section (Edwin Reynders)
% macec@bwk.kuleuven.be

gridfile = 'framegrid.asc';
slavefile = 'frameslave.asc';
beamfile = 'framebeam.asc';
file = strvcat('framedata.mat');
ext = strvcat('mat');
x = input2mcsignal(file,ext,100);
sens = [1;1;1;1];
ampl = [1;1;1;1];
db = [0;0;0;0];
types = cell_(strvcat('acc','acc','acc','acc'));
labels = cell_(strvcat('1','2','3','4'));
for par = 1:1, cnvdats{par} = volt2xa(x{par},sens',ampl',db',types',labels'); end
predats = cnvdats;
node_num = [5;4;3;2];
meas_dir = [0  0;0  0;0  0;0  0];
predat = predats{1};
chansel = [1  2  3  4];
refs = [1];
invar = ssi_data3(predat.tdata(:,chansel)',30,refs);
sysmat = sysmatcalc(invar,'ssi_data',chansel,[],refs,2:2:60);
[node_num,meas_dir,quants,chan_outin,ampl_outin] = ...
      identsel(node_num,meas_dir,predat.quantity',sysmat.chanselout,sysmat.chanselin);
allmodes = stable_propmodpar5(sysmat,predat.dt,chan_outin,ampl_outin,quants);
selnrs = [];
[modenr,yes,selnrs] = modfind(allmodes,2.7664,16,selnrs);
[modenr,yes,selnrs] = modfind(allmodes,7.9459,16,selnrs);
[modenr,yes,selnrs] = modfind(allmodes,12.135,16,selnrs);
[modenr,yes,selnrs] = modfind(allmodes,15.1606,16,selnrs);
stabmodes = stabpick(allmodes,selnrs);
```

Figure 3.7: Example of a logfile: the determination of the modal characteristics in tutorial 1 with SSI-data/ref.

Chapter 4

Tutorials

In this chapter, three tutorials are provided that get you acquainted with MACEC. The first tutorial concerns an operational modal analysis of a frame structure from simulated data. The second tutorial concerns an experimental and a combined modal analysis of a highway bridge from measured data. In the third tutorial, the batch mode of MACEC is explored for processing data from a roving hammer test in an efficient way.

4.1 Operational modal analysis of a frame structure

4.1.1 Introduction

As a comparative test between the different ambient system identification methods, the simulated ambient data of a linear, time-invariant frame structure with four degrees of freedom are analyzed in this tutorial. The objective is to perform an operational modal analysis using the different OMA methods that are available in MACEC. The data contain the horizontal acceleration response at each floor of the structure to a stationary Gaussian white noise vector excitation (fig. 4.1), simulated in MATLAB with a simple mass-spring-damper model, using MACEC functions. The file with which the data have been generated is named `framedata_generation.m`. It can be found in the `tuturial1` directory. The structure is proportionally damped.

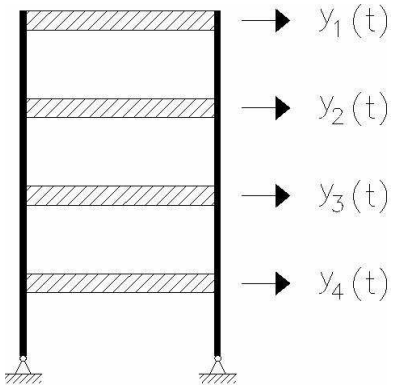


Figure 4.1: Tutorial 1: Frame structure

The simulated data are characterized by the following specifications:

- Four measurement channels contain four different horizontal acceleration signals;
- The data consist of 8192 time samples, obtained at a sampling frequency of 100 Hz.

4.1.2 Building the geometry

The first thing to do is to construct the grid of measurement nodes, and to connect them into a beam model of the structure for visualization. This consists of the following steps:

1. Start MATLAB;
2. Change the MATLAB working directory to the directory you want to run the tutorial in, for instance:

   ```
   C:\Program Files\MATLAB\R2010b\toolbox\spice\tutorials\tutorial1
   ```

3. In the Command Window of MATLAB, type *macec* and then press ENTER. Now the MACEC GUI appears;
4. in the MACEC GUI, press the MAKE GRID FILE button (fig. 4.2);

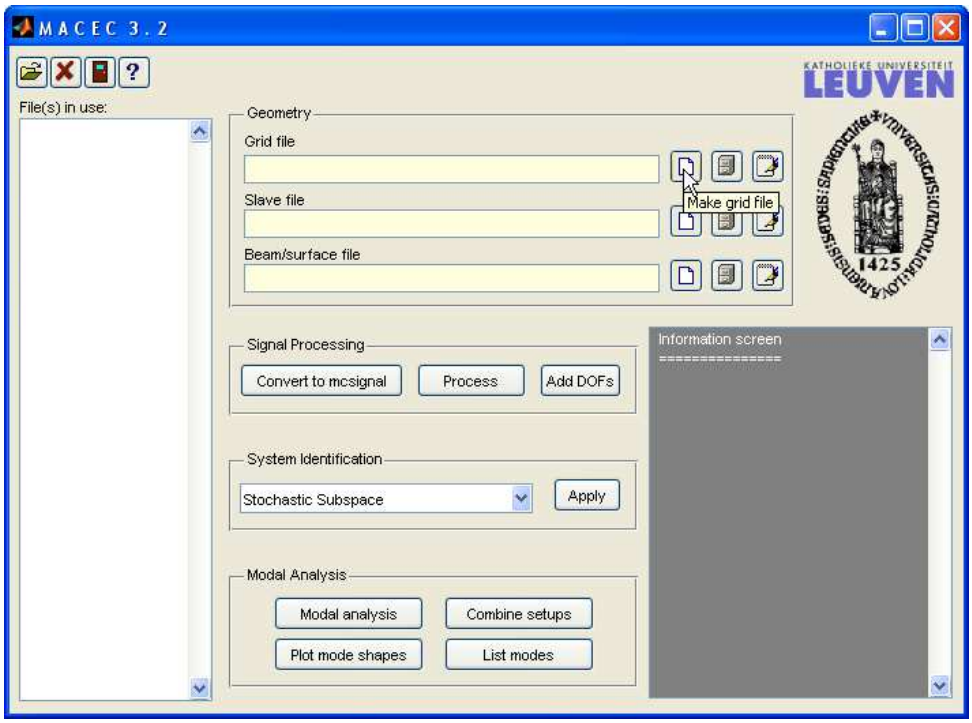


Figure 4.2: Tutorial 1: Starting the GUI for making a grid file.

5. Now the DEFINE / EDIT MEASUREMENT GRID window appears. Enter the following node numbers and coordinates and then press the ADD button (fig. 4.3):

node no.	X	Y	Z
1	0	0	0
2	0	0	2
3	0	0	4
4	0	0	6
5	0	0	8
6	5	0	0
7	5	0	2
8	5	0	4
9	5	0	6
10	5	0	8

Each added node is automatically plotted in the right figure of the DEFINE / EDIT MEASUREMENT GRID window (fig. 4.3). This enables you to check visually if the node numbers and the corresponding coordinates have been entered correctly. If you want to edit or save the figure with node numbers and coordinates, press the EDIT/SAVE FIGURE button.

Note: it is also possible to add multiple nodes at the same time by using standard MATLAB commands (fig. 4.4);

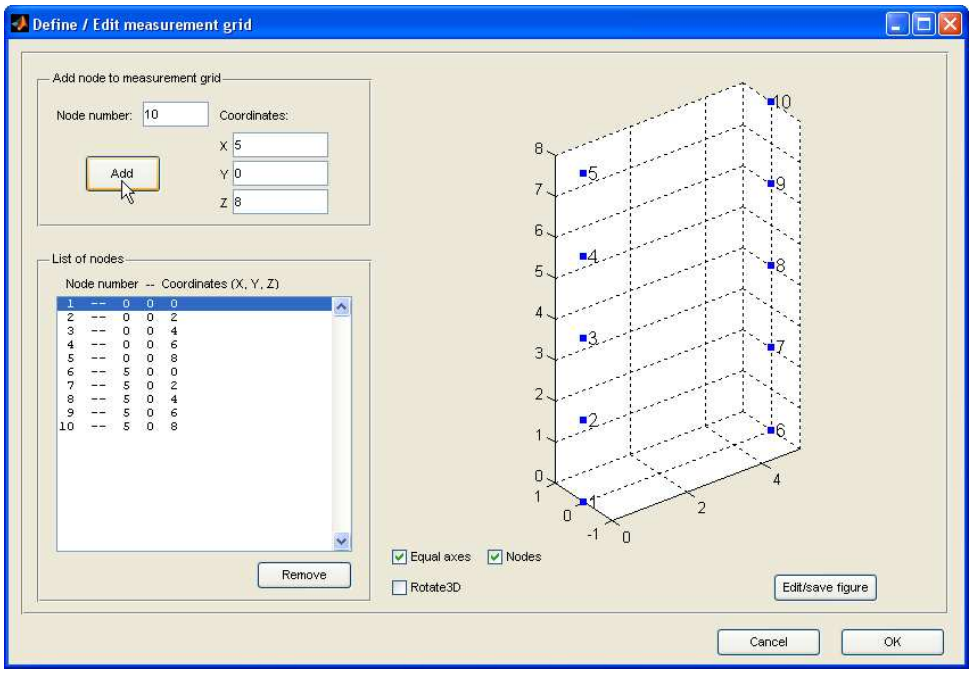


Figure 4.3: Tutorial 1: Entering node numbers and coordinates for the creation of the grid file.

6. Now you are ready to save the grid file. Press OK in the DEFINE / EDIT MEASUREMENT GRID window and choose a name and a directory (for instance `framegrid.asc` in the `tutorial1` di-

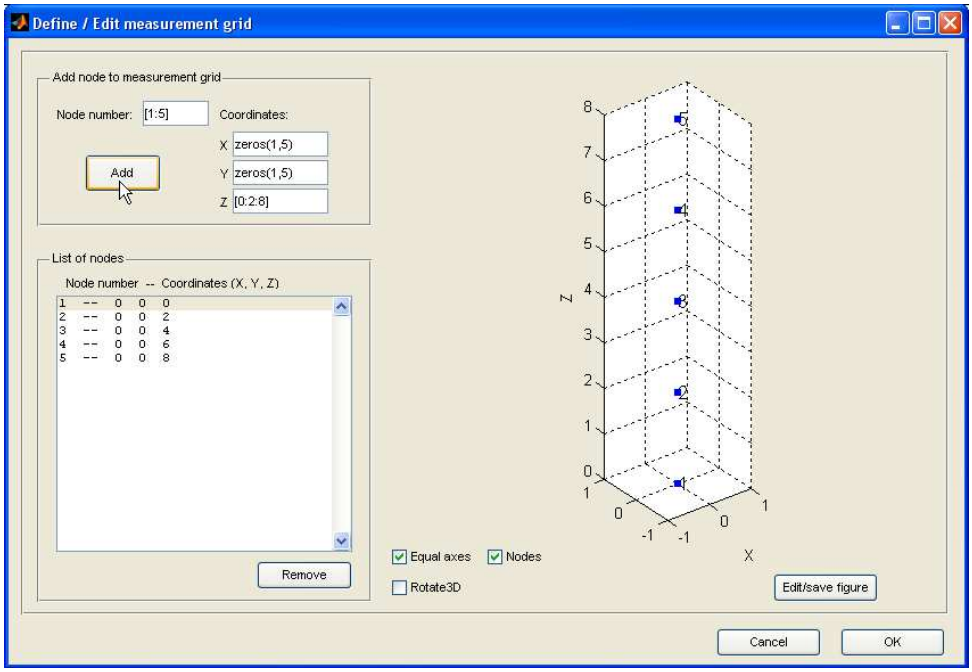


Figure 4.4: Tutorial 1: Entering multiple node numbers and coordinates at the same time.

rectory). The DEFINE / EDIT MEASUREMENT GRID window closes and you return automatically to the MACEC main window. In this window, the path name of the grid file you just created is now filled in in the GRID FILE command line of the GEOMETRY section.

Note: The grid file is saved in ASCII format so that it can also be created with a text editor instead of with the MACEC GUI. This facilitates the creation of the grid file when the nodal information is available in ASCII format (for instance from Microsoft® Excel®). The grid file is selected by typing its path in the GRID FILE *command line of the MACEC main window or by using the* SELECT GRID FILE *button of the MACEC main window*;

7. Because only the horizontal accelerations at one side of the frame structure have been simulated, a slaving procedure is needed to define the mode shapes at the other side of the frame. Therefore, click the MAKE SLAVE FILE button in the MACEC main window. The DEFINE / EDIT SLAVE DOFS window now appears (fig. 4.5). At the right, the nodes defined in the grid file are shown for your convenience.

   How should the DOFs be linked? It can be assumed that the horizontal beams of the frame structure are very rigid in the horizontal direction. Furthermore, nodes 1 and 6 are not measured, but it can be assumed that they have zero displacement. Therefore, the following DOFs should be coupled:

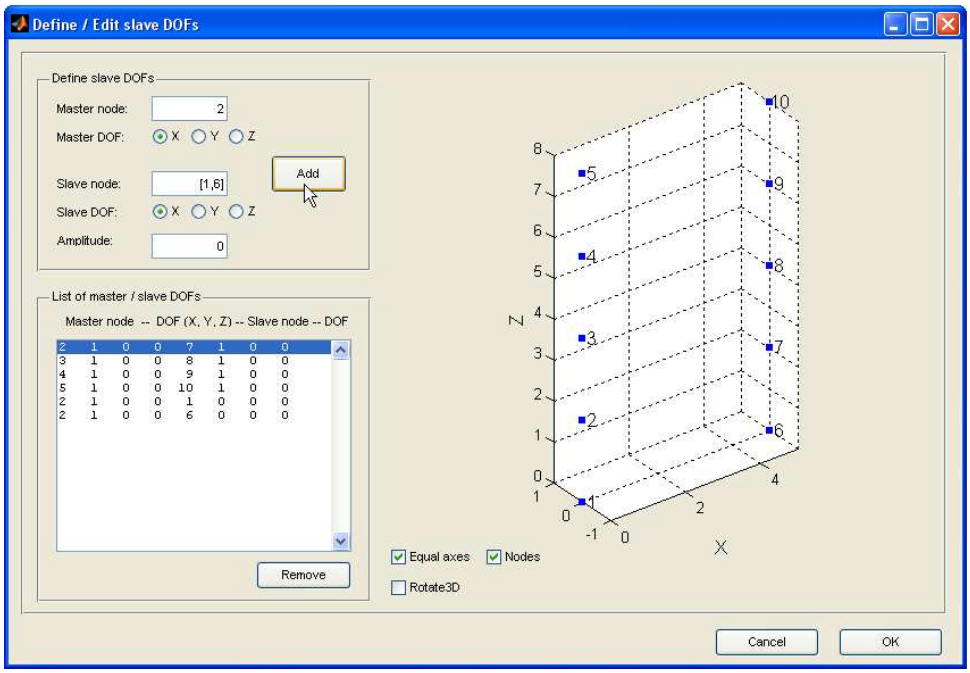


Figure 4.5: Tutorial 1: Linking master DOFs and slave DOFs using the GUI.

master node	master DOF	slave node	slave DOF	amplification factor
2	X	7	X	1
3	X	8	X	1
4	X	9	X	1
5	X	10	X	1
2	X	1	X	0
2	X	6	X	0

These DOFs can be coupled one by one, or alternatively at once using MATLAB vector notation. We will follow the second alternative, as it is more convenient here. In the MASTER NODE field, fill in

[2:5]

As the MASTER DOF, choose X. In the SLAVE NODE field, fill in

[7:10]

As the SLAVE DOF, choose also X. In the AMPLITUDE field, fill in

1

and then press the ADD button. The coupled DOFs are now added to the list at the left bottom (fig. 4.5). To plot also the nodes with zero displacement, fill in

2

as MASTER NODE, choose X as MASTER DOF, fill in

[1,6]

in the SLAVE NODE field and choose X as SLAVE DOF. In the AMPLITUDE field, fill in

0

and then press the ADD button;

8. Now you are ready to save the slave file. Press OK in the DEFINE / EDIT SLAVE DOFS window and choose a name and a directory (for instance `frameslave.asc` in the `tutorial1` directory). The DEFINE / EDIT SLAVE DOFS window closes and you return automatically to the MACEC main window. In this window, the path name of the slave file you just created is now filled in in the SLAVE FILE command line of the GEOMETRY section.

   Note: The slave file is saved in ASCII format so that it can also be created with a text editor instead of with the MACEC GUI. This facilitates the creation of the slave file when the slaving information is available in ASCII format (for instance from Microsoft Excel). The slave file is selected by typing its path in the SLAVE FILE *command line of the MACEC main window or by using the* SELECT SLAVE FILE *button of the MACEC main window*;

9. Now that the measurement grid has been defined and the necessary DOFs have been coupled in a slave file, the only geometrical information that is missing are the links between the measurement points. They are only defined for visualization purposes. In the MACEC main window, press the CREATE BEAM OR SURFACE FILE button. A new window, entitled DEFINE / EDIT BEAMS OR SURFACES appears.

   MACEC offers two alternatives for the visualization of the link information between the measurement points:

   - beams, which are defined by two node numbers (lines);
   - surfaces, which are defined by three node numbers (triangles).

   As the structure under study is a frame structure, beams are the natural choice for the visualization. Again, MACEC offers the possibility to accelerate the definition of the beams by using MATLAB vector notation. To connect the nodes at the left columns of the frame structure, enter

   [1:4]

   in the NODE 1 field and

   [2:5]

   in the NODE 2 field. If you then click the ADD button, the defined beams are added to the list at the left bottom and to the figure at the right. Now follow the same procedure for connecting the right columns and the horizontal beams of the frame structure. The result should look like in fig. 4.6;

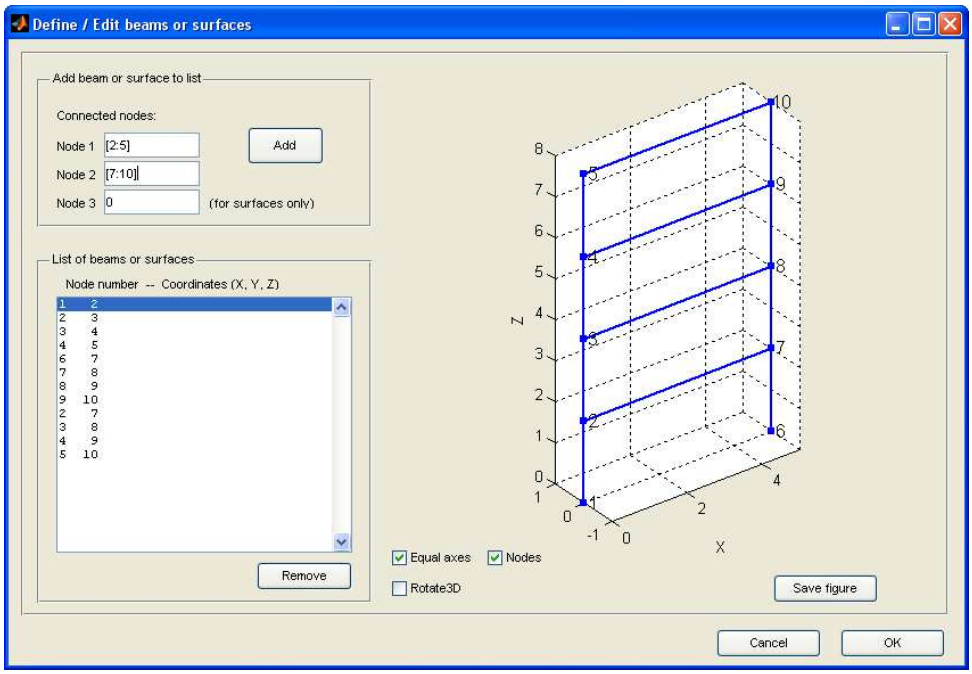


Figure 4.6: Tutorial 1: Creation of links between the measurement points for visualization.

10. Now you are ready to save the beam file. Press OK in the DEFINE / EDIT BEAMS OR SURFACES GRID window and choose a name and a directory (for instance `framebeam.asc` in the `tutorial1` directory). The DEFINE / EDIT BEAMS OR SURFACES GRID window closes and you return automatically to the MACEC main window. In this window, the path name of the beam file you just created is now filled in in the BEAM/SURFACE FILE command line of the GEOMETRY section.

    Note: The beam file is saved in ASCII format so that it can also be created with a text editor instead of with the MACEC GUI. This facilitates the creation of the beam file when the slaving information is available in ASCII format (for instance from Microsoft Excel). The beam file is selected by typing its path in the BEAM/SURFACE FILE *command line of the MACEC main window or by using the* SELECT BEAM OR SURFACE FILE *button of the MACEC main window*;

4.1.3 Processing the measured signals

The geometry of the frame structure is now defined, so the signal processing part can start.

11. First, select the file with the simulated measurement data by pushing the SELECT NEW DATA... button in the MACEC main window (fig. 4.7). Choose the file `framedata.mat` in the `tuturial1` directory;

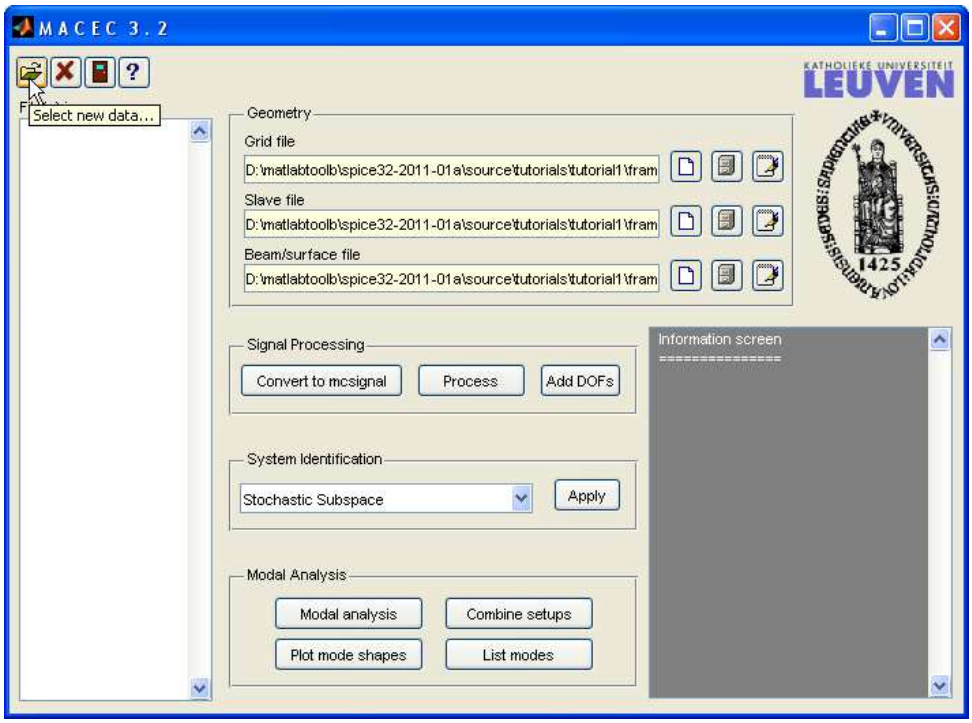


Figure 4.7: Tutorial 1: Specify a file's name and directory.

12. Now the name of this file appears in the FILE(S) IN USE list in the MACEC main window. Click on the filename and press the CONVERT TO MCSIGNAL button in the SIGNAL PROCESSING section. A dialog box appears which asks for the sampling frequency (fig. 4.8). As the sampling frequency is 100 Hz, enter

100

and press OK;

13. Now, a window opens which guides you through the conversion of the measurement data to an MCSIGNAL object. Each channel can be labeled for later reference. For instance, a good label for the first channel is "5X" since this channel contains the acceleration measured at node five in the X-direction. To provide this label, fill in

5X

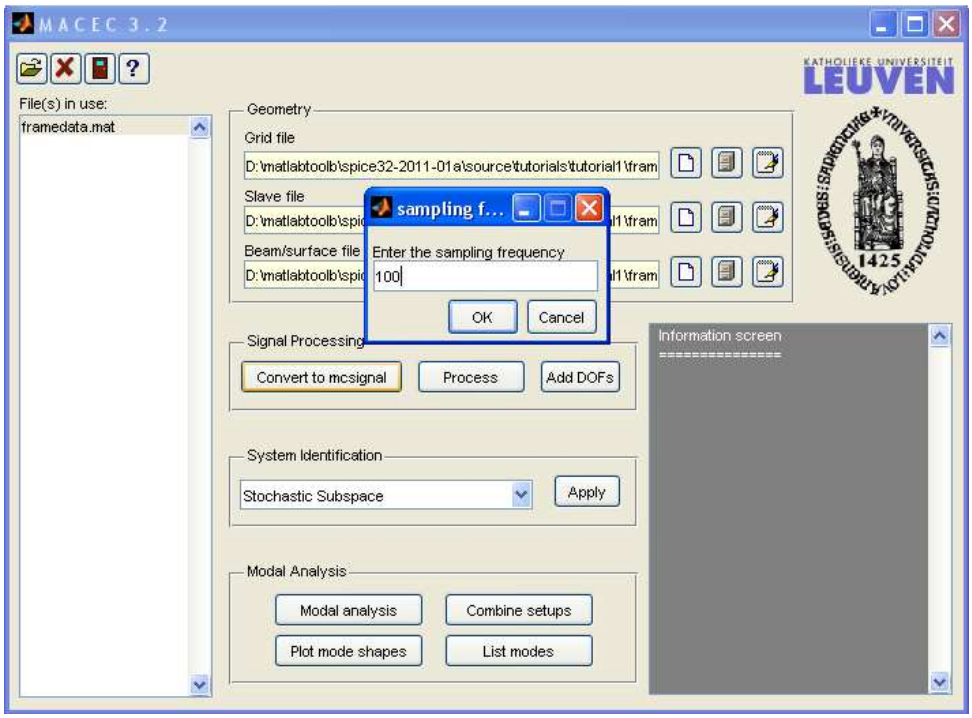


Figure 4.8: Tutorial 1: If the measurement data is available in MAT format, the sampling frequency must be provided.

in the LABEL field (fig. 4.9). As the measurement data have the physical meaning of accelerations, you don't need to change the DATA TYPE. The data are given in m/s^2, so you can leave the MEASUREMENT UNITS, the SENSITIVITY and the AMPLIFICATION fields to their default values;

14. In the same way, you can provide the labels for the other 3 channels. Afterwards, press the OK button (fig. 4.9). Now, the MCSIGNAL object will be saved in a `*.mat` file, for which you have to provide a proper name like for instance

    `framedata_conv.mat`

    You can also specify a directory, different from the current working directoy;

15. After you have saved the MCSIGNAL object, MACEC automatically returns to the main window, where the file `framedata_conv.mat` has now been added to the FILE(S) IN USE section. The next step is the actual processing of the measurement signals. In order to do so, select the `framedata_conv.mat` file in the FILE(S) IN USE section and press the PROCESS button in the SIGNAL PROCESSING section. The PREPROCESS MCSIGNAL OBJECT window opens (fig. 4.10). You can have a look at the time history and the frequency content of the different signals. As the simulated measurement data in this case need no further preprocessing, just press the OK button (fig. 4.10) and save the preprocessed MCSIGNAL object as[1]

    `framedata_proc.mat`

[1] Please note that the SAVE and RESUME buttons do not save the preprocessed MCSIGNAL object to a file, but are used to intermediately save the processed data and to resume from the last save, respectively.

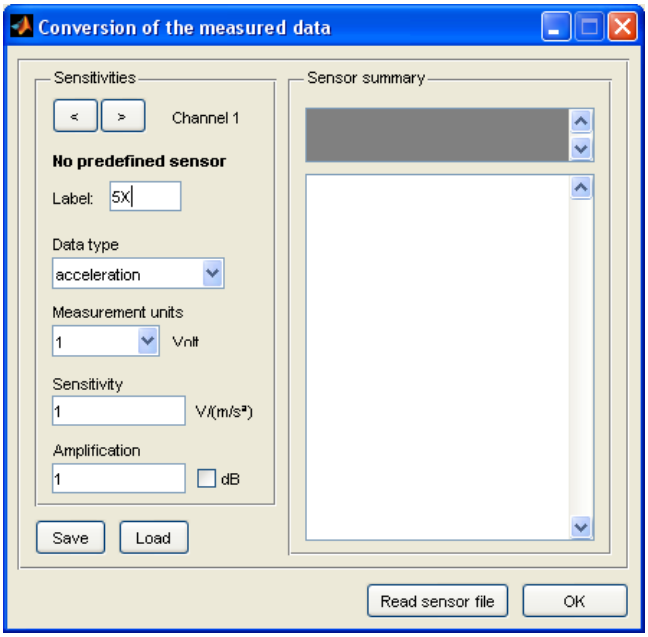


Figure 4.9: Tutorial 1: Conversion of the measurement data to an MCSIGNAL object.

16. A last thing to do before the System Identification can start is the the coupling between the measurement nodes defined in the grid file and the measurement channels. Select `framedata_proc.mat` in the MACEC main window and press the ADD DOFS button. Now the ADD CHANNEL SPECIFICATIONS window opens (fig. 4.11). For each channel, you have to provide the corresponding grid node number and measurement direction. Note that the channel label turns out to be a useful feature here. If you are ready, press SAVE to add the channel information to the `*.mat` file with the MCSIGNAL object, `framedata_proc.mat`.

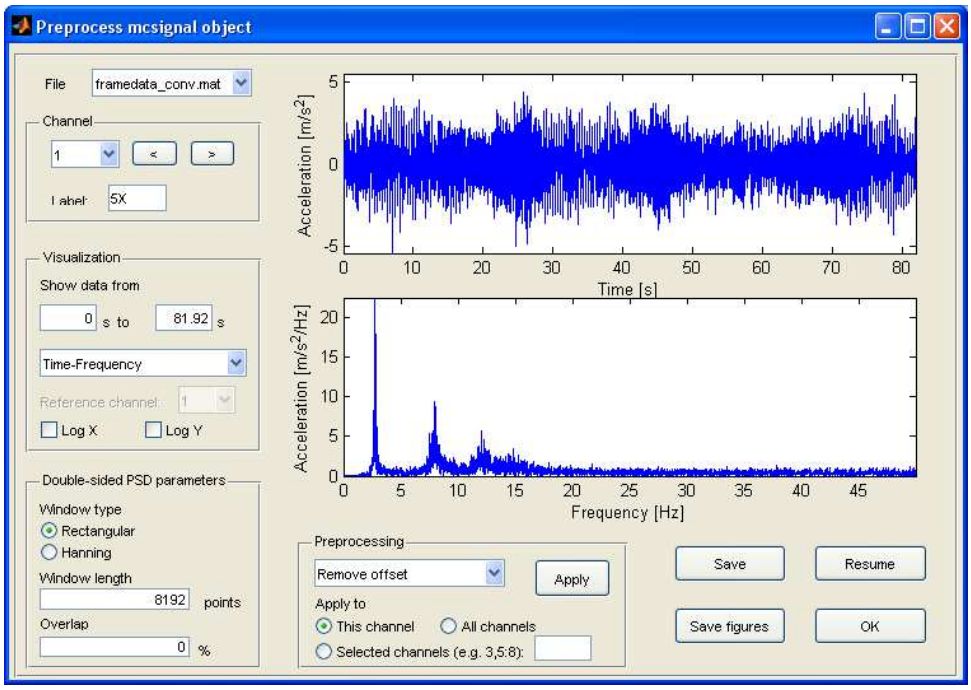


Figure 4.10: Tutorial 1: Preprocessing the mcsignal object.

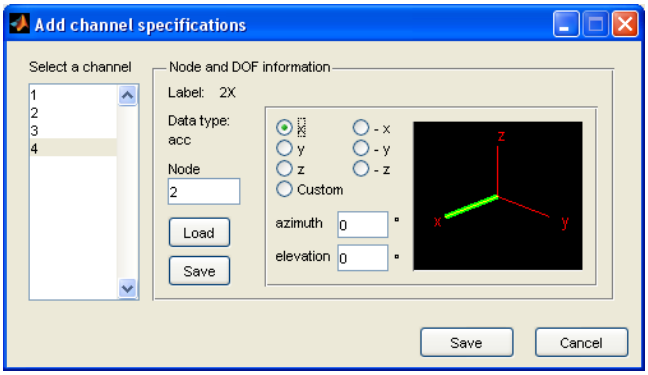


Figure 4.11: Tutorial 1: Coupling between the measurement nodes in the grid file and the measurement channels.

4.1.4 Identification of a linear system model

Now you are ready to perform system identification [11]. As you have only output data at your disposal, the only system identification methods you can use are output-only or stochastic system identification methods. MACEC contains four of these methods:

- Nonparametric PSD$^+$ estimation using the correlogram or periodogram approaches;
- Reference-based data-driven stochastic subspace identification (SSI-data/ref);
- Reference-based covariance-driven stochastic subspace identification (SSI-cov/ref);
- Operational poly-reference least squares complex frequency domain identification (pLSCF).

The purpose of this section is to make you acquainted with these different methods, which have each there own advantages and disadvantages.

... using nonparametric PSD$^+$ estimation

17. Select the file `framedata_proc.mat` in the *File(s) in use* section of the MACEC main window, change the identification method in the SYSTEM IDENTIFICATION section to NONPARAMETRIC (FOR PEAK PICKING OR CMIF/FDD) and press APPLY. The NONPARAMETRIC FRF / PSD+ ESTIMATION FOR PEAK PICKING OR CMIF/FDD window appears. As the data of the frame structure are output-only, the ANALYSIS TYPE should be set to STOCHASTIC. Please note that MACEC automatically suggests the right ANALYSIS TYPE based on the data type of the channels (in this case accelerations only).

18. The system model that will be identified is a Positive Power Spectral Density (PSD$^+$) matrix at discrete frequency lines. The Positive Power Spectral Densitiy $S^+_{XY}(\omega)$ between two channels X and Y is defined as the Fourier transform of the *positive* lags of the cross-correlation function $r_{XY}(t)$ between these two channels [2]:

$$S^+_{XY}(\omega) = \int_{-\infty}^{\infty} r_{XY}(t)e^{-j\omega t}u(t)dt = \int_{0}^{\infty} r_{XY}(t)e^{-j\omega t}dt$$

where $u(t)$ is the unit step function (equal to zero for negative values of t and equal to one for positive values of t).

Before starting the PSD$^+$ estimation, you should first select which outputs will be used for the identification in the SELECT CHANNELS section. As all channels contain valid output data, they should all be selected in the OUTPUTS column. Please note that also here, MACEC automatically suggests the right choice depending on the data type of each channel. It is also possible to work with reference channels, but we will not make use of this possibility now. Therefore, fill in

1:4

in the REFERENCES field of the POSITIVE POWER SPECTRAL DENSITY CORRELOGRAM ESTIMATION so that all channels are selected as reference channels and press APPLY (fig. 4.12);

19. The PSD+ ESTIMATION window opens (fig. 4.13). The user has the choice between two different methods for the estimation of the positive power spectral densities: the correlogram and the periodogram method.

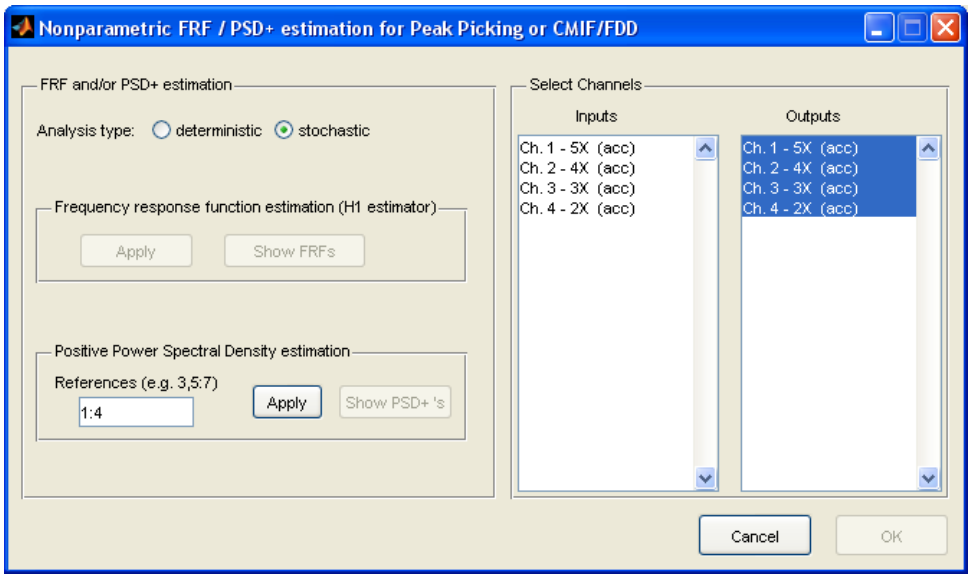


Figure 4.12: Tutorial 1: Nonparametric PSD$^+$ estimation.

The correlogram method calculates the PSD+ by first estimating the correlation function through averaging and then taking the discrete Fourier transform. For larger time lag values, there are less data points available for averaging, which makes that the correlation estimates are less accurate. This can also be verified for this case. Select the SHOW CORRELATION option in the PLOT OPTIONS part, leave all other settings to their default values, and press CALCULATE AND SHOW ESTIMATES (fig. 4.13). From the plot, you can see that the value of the autocorrelation estimate decreases between the time lags of 0 and 12s, as expected. However, when the lag is larger than 12s, the estimates increase again, which indicates their inaccuracy. Therefore, fill in

1024

for the NUMBER OF TIME LAGS and press CALCULATE AND SHOW ESTIMATES. If you select now SHOW PSD+ in PLOT OPTIONS, you notice that the new estimate is much smoother than the previous one. After unchecking the SHOW ESTIMATION HISTORY box, fig. 4.14 is obtained. It should be kept in mind that, for short data sequences, the number of useful correlations might be low, and leakage errors might become important [5].

20. The periodogram method calculates the PSD+ by dividing the available raw time data into N_b blocks, doubling the length of each block by adding zeros, taking the Fourier transform for each block, multiplying for each frequency all outputs with the Hermitian transpose of the reference outputs, and averaging the result over all blocks [2]. Note that in this way, the number of blocks is an upper bound for the rank of the PSD+ matrix.

    In order to reduce the variance error of the perdiodogram estimate, one can take the inverse Fourier Transform of it, apply a time window to the resulting correlation estimate, and take the Fourier transform again [2, p. 58]. This approach usually yields much better results than taking an excessive number of averages, resulting in short data blocks and as a consequence large leakage errors. However, it results in biased estimates for the modal participation factors, which need only to be compensated for when the contributions of the different modes to the PSD+ is of interest [2].

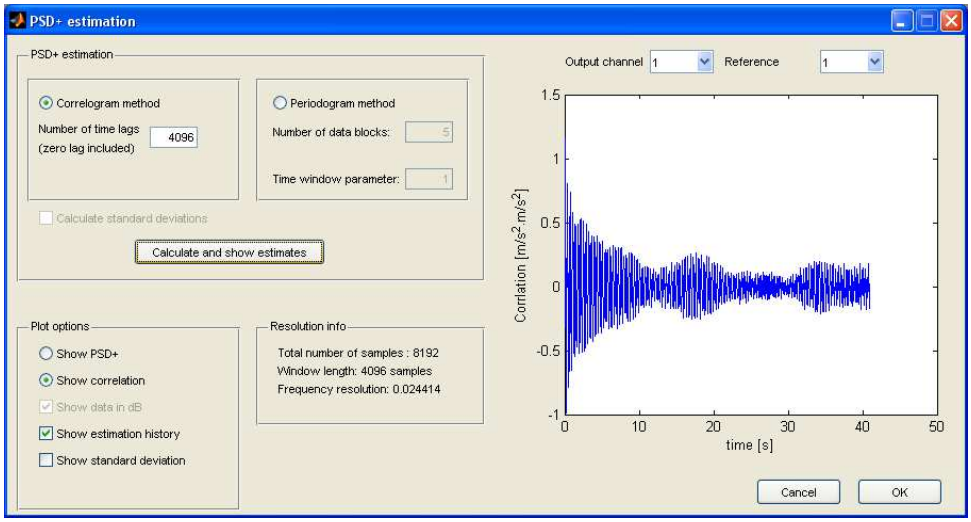


Figure 4.13: Tutorial 1: Autocorrelation estimation before PSD+ estimation.

In the PERIODOGRAM METHOD section, the user has to provide two values: the number of blocks that the raw time data will be divided in and the time window parameter m, which equals the ratio between the length of one block N_b and the length of the rectangular time window used for the noise reduction N_w:

$$m = \frac{N_b}{N_w}$$

As the total number of time samples we have at our disposal is 8192 (this can be verified from the RESOLUTION INFO box at the left bottom), choose

8

as the NUMBER OF DATA BLOCKS, so that each block will contain 1024 samples. To appreciate the effect of the TIME WINDOW PARAMETER, first leave it to its default value

1,

check CALCULATE STANDARD DEVIATIONS and SHOW STANDARD DEVIATION, press CALCULATE AND SHOW ESTIMATES, and uncheck SHOW ESTIMATION HISTORY. The estimated PSD+ curve is now shown in blue and the estimated standard deviation is shown in grey. Now, change the TIME WINDOW PARAMETER to

2

and press again CALCULATE AND SHOW ESTIMATES. Now you see that the PSD+ curve is much smoother than for $m = 1$, which indicates that the noise level has been seriously reduced (fig. 4.15). This is also reflected in the estimated standard deviation, which has decreased. However, there's a price to pay: the frequency resolution has increased with a factor 2 to 0.2 Hz (this can be verified from the RESOLUTION INFO box at the left bottom). Therefore, do not increase m any further but press OK (fig. 4.15). The last PSD+ estimate is passed to the NONPARAMETRIC FRF / PSD+ ESTIMATION FOR PEAK PICKING OR CMIF/FDD window.

21. In this window, press OK and save the models in the following `*.mat` file:

`framedata_nonpar.mat`

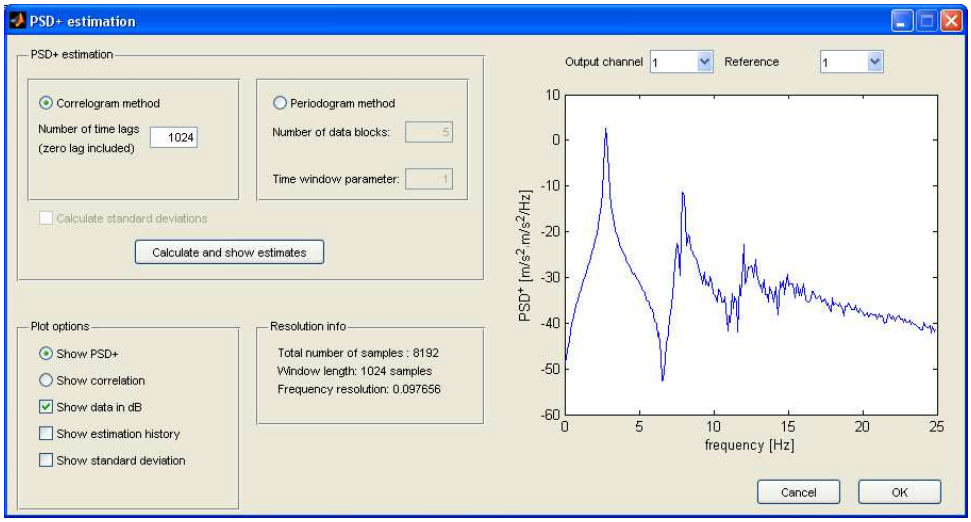


Figure 4.14: Tutorial 1: PSD+ estimation with the correlogram method.

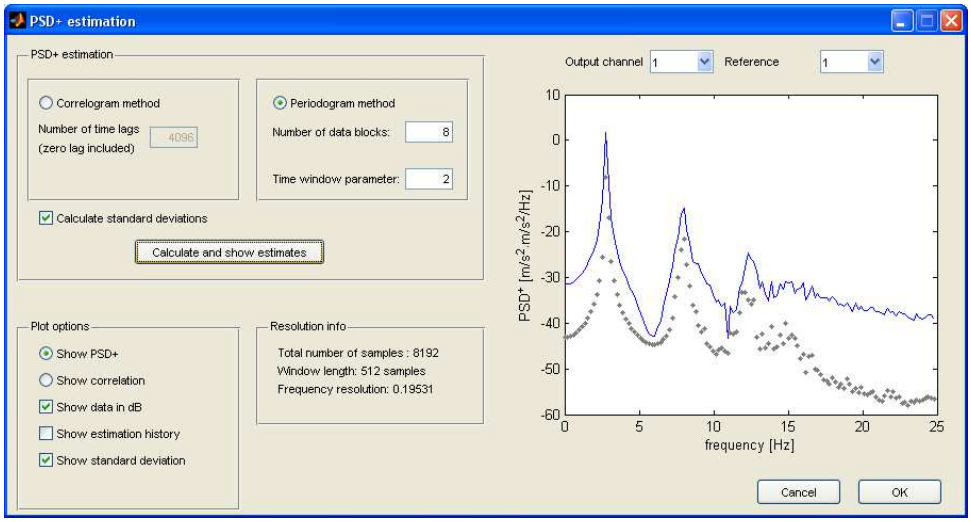


Figure 4.15: Tutorial 1: PSD+ estimation with the periodogram method.

... using SSI-data

22. Select the file `framedata_proc.mat` in the FILE(S) IN USE section of the MACEC main window, don't change the identification method in the SYSTEM IDENTIFICATION section and press APPLY. Now the STOCHASTIC SUBSPACE IDENTIFICATION window appears. Leave the ALGORITHM SELECTION on DATA-DRIVEN[2]. In the EXPECTED SYSTEM ORDER field, you have to specify the theoretical order of the system description that you need. This order equals two times the expected number of modes that are present in the data. As the frame structure is a 4 DOF structure, the expected system order equals $2 \cdot 4 = 8$. Now press the >> button and notice that MACEC proposes the number

[2]The theory behind the SSI-data algorithm is not explained in this manual. The interested reader is referred to *[7, 17]* for more information.

as half the number of block rows i in the SSI-data algorithm[3]. However, this proposed value can be viewed as a minimal value for i in the SSI-data algorithm, since it is known that higher values of i usually yield more accurate system estimates[4]. Furthermore, it offers the possibility to choose higher stabilization orders in the stabilization diagram. So choose

30

as the value for i and press the CALCULATE QR + SVD button to start the construction of the Hankel matrix of measurement data and the QR and SVD steps of the SSI algorithm (fig. 4.16).

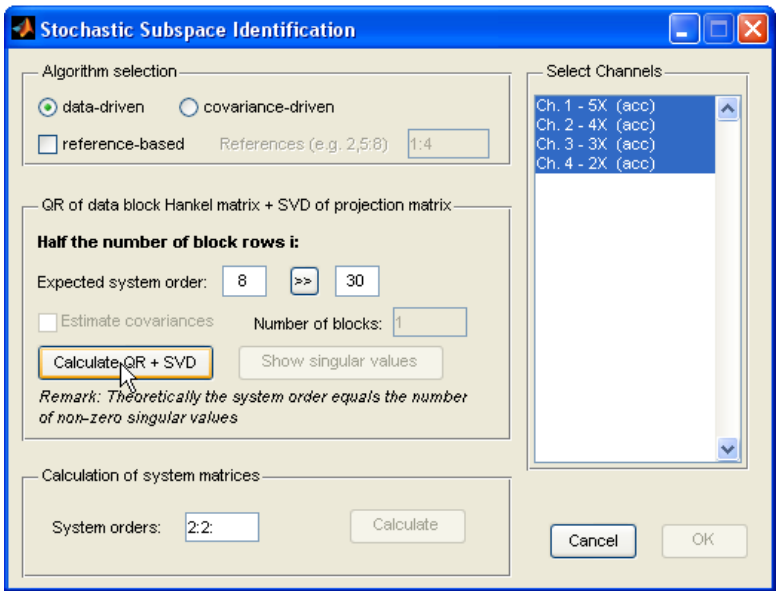


Figure 4.16: Tutorial 1: Data-driven SSI.

23. After the QR and SVD steps of the SSI algorithm have been performed, the real system order can be estimated by looking at the singular values calculated from the SVD step. For noiseless data, the system order equals the number of nonzero singular values. For noisy data, the noise causes some singular values to be different of zero. However, their values are usually very low. After pressing the SHOW SINGULAR VALUES button, a standard MATLAB figure appears which shows the singular values in decreasing order of magnitude (fig. 4.17). As can be seen from the figure, the first eight singular values are clearly larger than the other ones, which indicates that a model order of 8 could be able to describe the system dynamics quite well. The other singular values are not exactly zero, but it can be seen that choosing a model order higher than 20 scarcely influences the dynamics of the identified system.

24. We know that the true system order equals 8. However, from the above discussion, it is clear that, due to the noise which is present in the data, the identified system description contains

[3]The suggested value of i is calculated by multiplying the expected system order with the factor 8 and dividing the result by the number of reference channels. Please note that in MACEC 2.0, a factor 6 was used instead of 8. Due to the increased computation power and memory since the development of MACEC 2.0, it was possible to increase this factor for standard use in the MACEC 3.0 and later versions.

[4]Please consult [12] for a discussion on how the choice of i is connected to the relationship between the Nyquist frequency and the lowest frequency of interest.

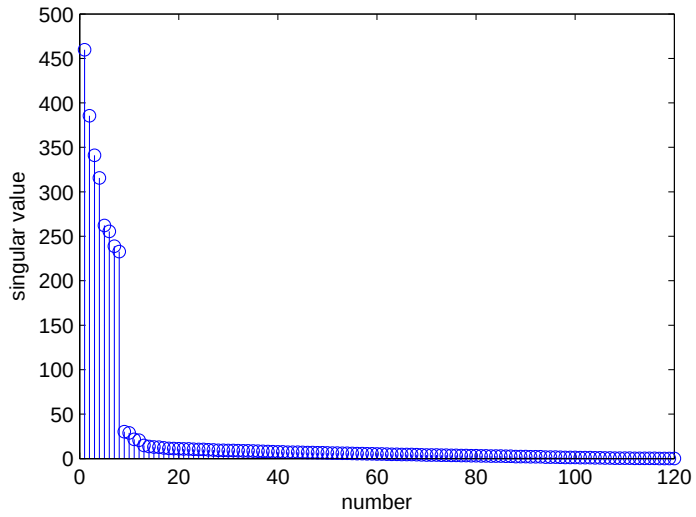


Figure 4.17: Tutorial 1: Data-driven SSI: singular values.

both system and noise dynamics. A common approach in modal analysis is then to over-specify the model order, such that the true system modes (also called physical modes) are separated from the noise modes (also called mathematical modes). This separation is performed manually in a stabilization diagram (see further). To create the stabilization diagram, models of increasing order need to be identified. From fig. 4.17, one knows that the maximal possible system order equals 120[5]. However, as the physical modes should stabilize from a model order between 8 and 12, we choose to construct the stabilization diagram only up to a model order of 60. So, in the System orders field, fill in

2:2:60

and press the Calculate button. System orders between 2 and 60 are now calculated in increasing steps of 2.

25. After the calculation is finished, press the OK button and save the identified system matrices in a `*.mat` file, for instance

`framedata_ssi_data.mat`

MACEC now returns to the main window, where the file has been added to the File(s) in use section.

... using SSI-data/ref

26. In the previous section, you have used the classical version of the SSI-algorithm as described in [17]. However, a faster reference-based version of this algorithm has been developed by Peeters and De Roeck [7], which is also more accurate if the channels with the highest SNR

[5]In general, the maximum possible system order equals i (which equals 30 in this case) multiplied by the number of reference channels (in this case equal to all 4 channels).

are selected as the reference channels[6]. To see if there is a noticeable difference with the classic version, repeat the SSI-identification again, but now mark the REFERENCE-BASED checkbox in the STOCHASTIC SUBSPACE IDENTIFICATION window and indicate that the first channel (which contains the horizontal displacement at the top) is the only reference channel (fig. 4.18). Note that system orders between 2 and 30 are now calculated in increasing steps of 2 (MACEC ignores automatically system orders that impossible, so system orders higher than 30 are not considered). Save the identified systems as

`framedata_ssi_data_ref.mat`

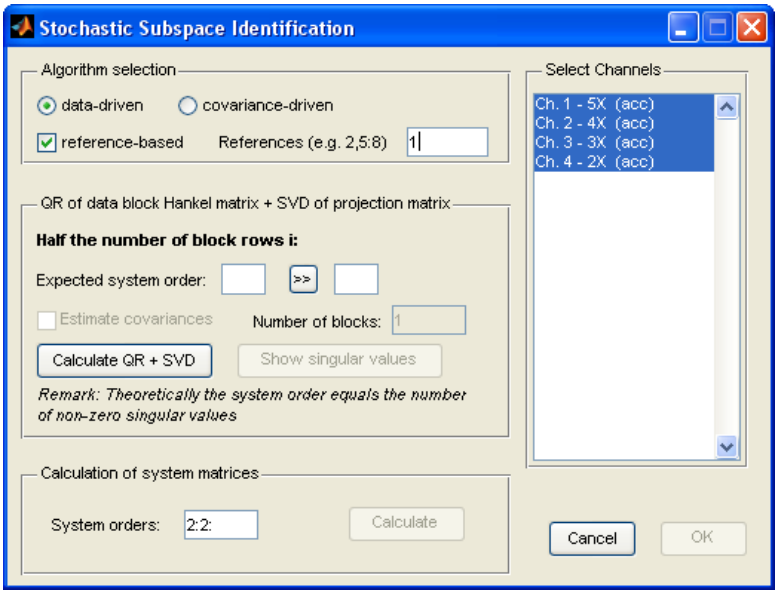


Figure 4.18: Tutorial 1: Reference-based data-driven SSI.

... using SSI-cov

27. Select the file `framedata_proc.mat` in the FILE(S) IN USE section of the MACEC main window, don't change the identification method in the SYSTEM IDENTIFICATION section and press APPLY. The STOCHASTIC SUBSPACE IDENTIFICATION window appears. Change the ALGORITHM SELECTION to COVARIANCE-DRIVEN[7]. As for the SSI-data analyses, choose

    30

    as the value for half the number of block rows.

28. An advantage of SSI-cov over SSI-data is that for SSI-cov, the variances of the identified system parameters (system matrices, poles, eigenfrequencies, damping ratios, mode shapes) can be estimated [14, 3]. In order to activate the variance computation, check the ESTIMATE COVARIANCES box. The NUMBER OF BLOCKS field, where you can specify the number of blocks in which the

[6] The theory behind the SSI-data/ref algorithm is not explained in this manual. The interested reader is referred to *[7]* for more information.

[7] The theory behind the SSI-cov algorithm is not explained in this manual. The interested reader is referred to *[1, 7, 14]* for more information.

raw time data will be divided for computing sample covariances of the output correlation matrices, is now enabled. The number of blocks should be chosen sufficiently high in order to increase the accuracy of the variance estimation, yet low enough to ensure that the different data blocks are still approximately statistically independent from each other. If the number of block rows i is already specified, MACEC proposes a default value for the NUMBER OF BLOCKS when the ESTIMATE COVARIANCE box is checked. Indeed, the value for NUMBER OF BLOCKS has been set to

32

In the first instance we can accept this value and press the CALCULATE QR + SVD button to start the identification algorithm (fig. 4.19).

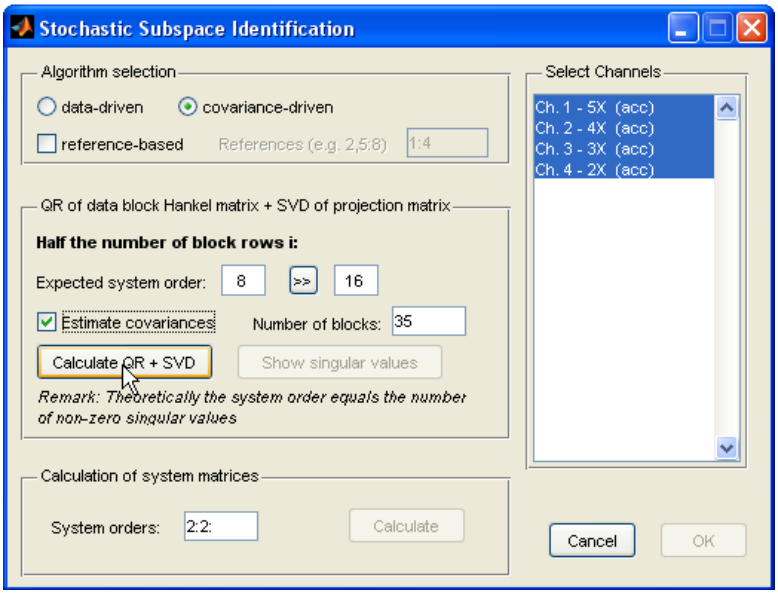


Figure 4.19: Tutorial 1: Covariance-driven SSI.

29. If you have a look at the singular values of the block Toeplitz matrix of covariance matrices, you can draw the same conclusions as for the SSI-data identification. Therefore, choose

    2:2:60

    as system order range and press the CALCULATE button. System orders between 2 and 60 are now calculated in increasing steps of 2.

30. After the calculation is finished, press the OK button and save the identified systems in a `*.mat` file, for instance

    `framedata_ssi_cov.mat`

    MACEC now returns to the main window, where the file has been added to the FILE(S) IN USE section.

... using operational pLSCF

31. Select the file `framedata_proc.mat` in the *File(s) in use* section of the MACEC main window, change the identification method in the SYSTEM IDENTIFICATION section to pLSCF and press APPLY. The pLSCF IDENTIFICATION window appears (fig. 4.20)[8];

32. The pLSCF algorithm is suitable for experimental or operational modal analysis, which corresponds to a deterministic or stochastic system identification, respectively. As the data of the frame structure are output-only data, the ANALYSIS TYPE in the FRF AND/OR PSD+ ESTIMATION section should be set to STOCHASTIC. Please note that MACEC automatically suggests the right ANALYSIS TYPE based on the data type of the channels (in this case accelerations only);

33. The pLSCF identification algorithm is a frequency-domain algorithm and therefore the first step of the algorithm is, for the stochastic case, the estimation of the matrix of Positive Power Spectral Densities (PSD+). Before starting the PSD+ estimation, you should first select which outputs will be used for the identification in the SELECT CHANNELS section. As all channels contain valid output data, they should all be selected in the OUTPUTS column. Please note that also here, MACEC automatically suggests the right choice depending on the data type of each channel. For the pLSCF algorithm, it is also possible to work with reference channels, but we will not make use of this possibility now. Therefore, fill in

    1:4

    in the REFERENCES field of the POSITIVE POWER SPECTRAL DENSITY ESTIMATION so that all channels are selected as reference channels and press APPLY (fig. 4.20);

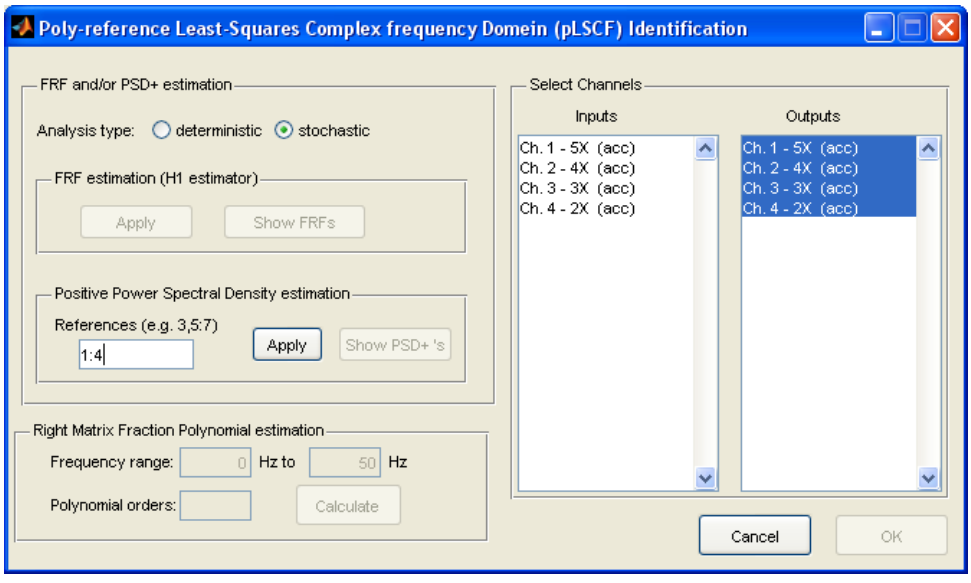


Figure 4.20: Tutorial 1: pLSCF.

34. The PSD+ ESTIMATION window opens (fig. 4.13). Perform the same actions as in paragraph 20 above (fig. 4.15) and press OK.

[8]The theory behind the pLSCF or Poly-reference Least Squares Complex Frequency Domain algorithm is not explained in this manual. The interested reader is referred to *[2]* for more information.

35. MACEC returns to the pLSCF Identification window (fig. 4.20), where the Right Matrix Fraction Polynomial estimation section has been enabled. Since the bandwidth of interest coincides with the available frequency band, don't change the default values in the Frequency range fields. In the Polynomials orders field, fill in

    1:15

    and press Calculate to start the calculation of Right Matrix Fraction Description (RMFD) models of orders increasing from 4 to 60 in steps of 4 (the system order equals the matrix polynomial order times the number of references and/or inputs).

36. After the RMFD models have been calculated, press OK and save the models in the following `*.mat` file:

    `framedata_pLSCF.mat`

4.1.5 Modal analysis of the identified system models

For the frame structure under consideration, five (series of) linear system models have now been identified:

1. a nonparametric PSD^+ model;
2. stochastic state-space models identified with SSI-data of orders $2, 4, 6, \ldots, 60$;
3. stochastic state-space models identified with SSI-data/ref of orders $2, 4, 6, \ldots, 30$;
4. stochastic state-space models identified with SSI-cov of orders $2, 4, 6, \ldots, 60$;
5. right matrix fraction description models identified with operational pLSCF of orders $4, 8, 12, \ldots, 60$.

The next step in the experimental determination of the modal parameters of the structure is the modal analysis of the identified system models. For each of the four series of parametric models, a stabilization diagram will be calculated and from this diagram, we will select the stable, physical system modes that we are looking for. But first, we will identify the modal parameters in a more intuitive, but rather rough way, starting from the identified nonparametric PSD^+ model.

... using nonparametric techniques

37. Select the `framedata_nonpar.mat` file in the File(s) in use section of the MACEC main window and press the Modal Analysis button in the Modal Analysis section. You are asked to select a modal analysis method. Choose Peak Picking. The averaged normalized power spectral density (ANPSD) is then computed and plotted in an interactive window. Since in this case, the modal parameters are well separated, they can be estimated by picking the peaks in the ANPSD plot[9] (fig. 4.21). Note that, after a peak has been selected, the corresponding mode shape is listed and plotted in the complex plane. After the modes have been selected, press OK and save the mode information as

[9]The theory behind the peak Picking method is not explained in this manual. The interested reader is referred to e.g. [8, 10] for more information.

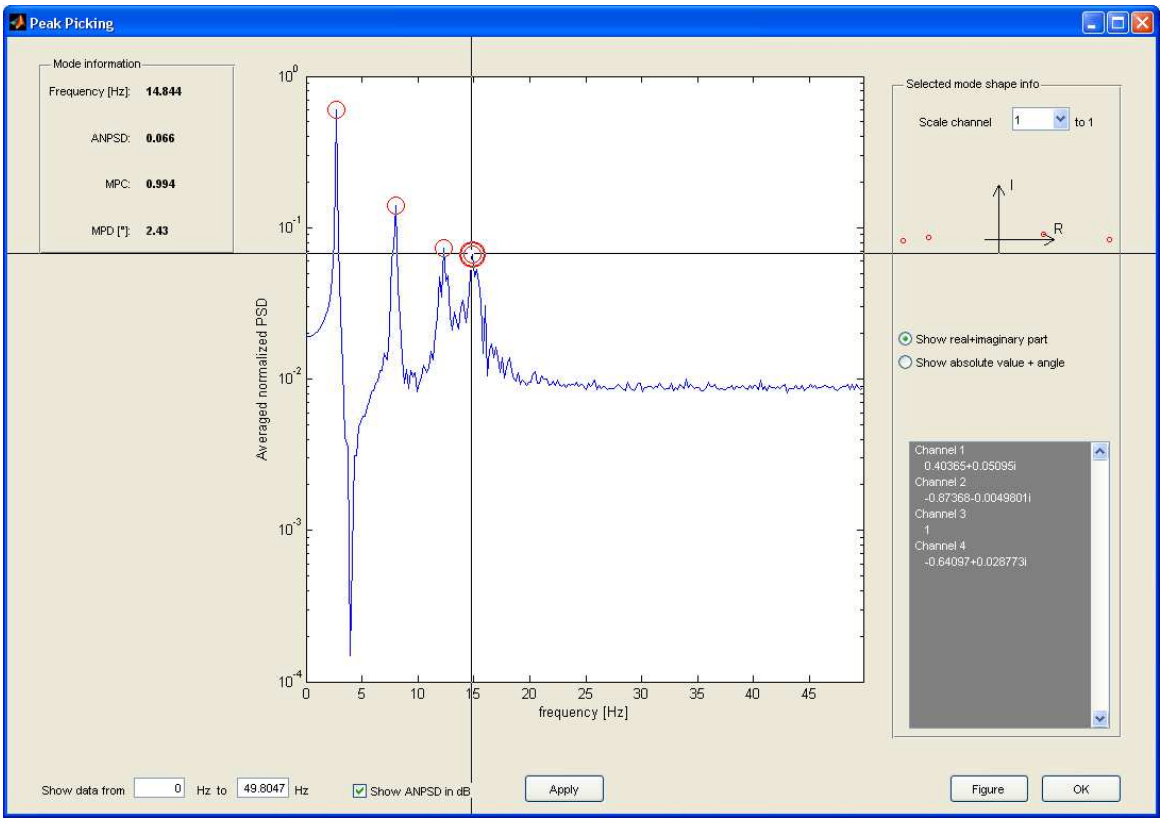


Figure 4.21: Tutorial 1: Selection of modes by Peak Picking.

`framedata_peakpick_modes.mat`

38. Now repeat the modal analysis with the `framedata_nonpar.mat` file, but select FDD as modal analysis method. The singular values of the PSD matrix, obtained by adding the estimated PSD^+ and its complex conjugate, are then computed and plotted in an interactive window. The modal parameters can be estimated by picking the peaks in the highest singular value(s)[10] (fig. 4.22). After the modes have been selected, press OK and save the mode information as

`framedata_fdd_modes.mat`

... using the stabilization diagram

39. Select the `framedata_ssi_data.mat` file in the FILE(S) IN USE section of the MACEC main window and press the MODAL ANALYSIS button in the MODAL ANALYSIS section. The modal parameters are calculated automatically and a stabilization diagram is created (fig. 4.24). Notice that, thanks to the preset values for the stabilization criteria, the diagram looks very clear.

40. Now tick the SHOW PSD+/FRFS checkbox and press CALCULATE. A small window opens (fig. 4.23). Leave all settings to their default values and press OK. The sum of all Positive Power Spectral Densities is now calculated and MACEC returns to the stabilization diagram. If

[10] For well-separated modes, as in this case, only the highest singular value should be considered. For closely spaced modes, however, the other singular values should be considered as well. The theory behind the CMIF/FDD method is not explained in this manual. The interested reader is referred to [16] for more information.

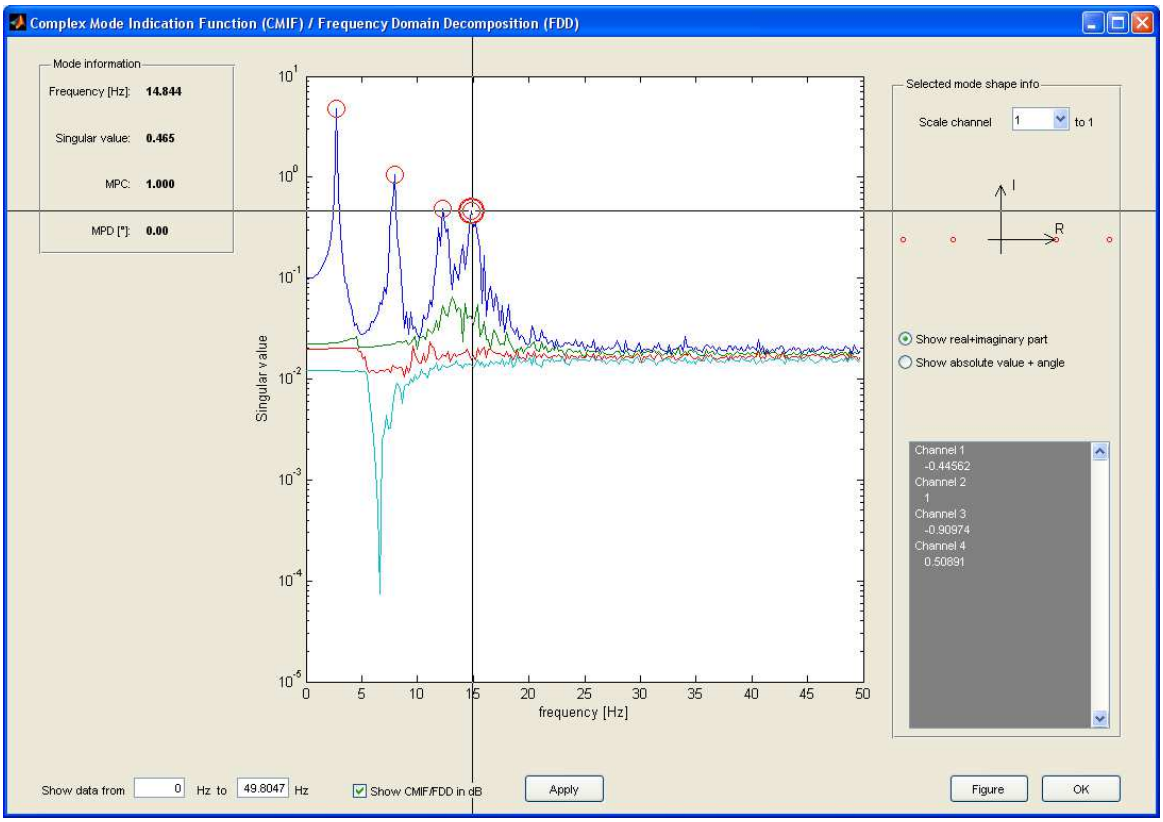


Figure 4.22: Tutorial 1: Selection of modes by Frequency Domain Decomposition.

you then press Apply, the PSD+ sum is added to the diagram to help you select the physical modes. Since the four lines of stable modes and the peaks in the PSD^+ sum all lie below 25 Hz, show only the data in the $0 - 25$ Hz bandwidth by filling out the relevant fields at the bottom and pressing Apply. Choose for instance the four stable modes at model order 52 by clicking on the modes in the diagram. After each mode has been clicked, the corresponding mode shape is plotted in the complex plane at the right-hand side of the stabilization diagram (fig. 4.24). This can be used as a tool to check the signature of each mode.

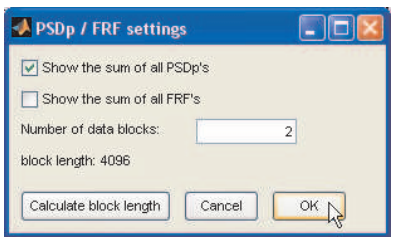


Figure 4.23: Tutorial 1: Calculation of Positive Power Spectral Densities for the stabilization diagram.

41. After the modes have been selected, press OK and save the mode information as

`framedata_ssi_data_modes.mat`

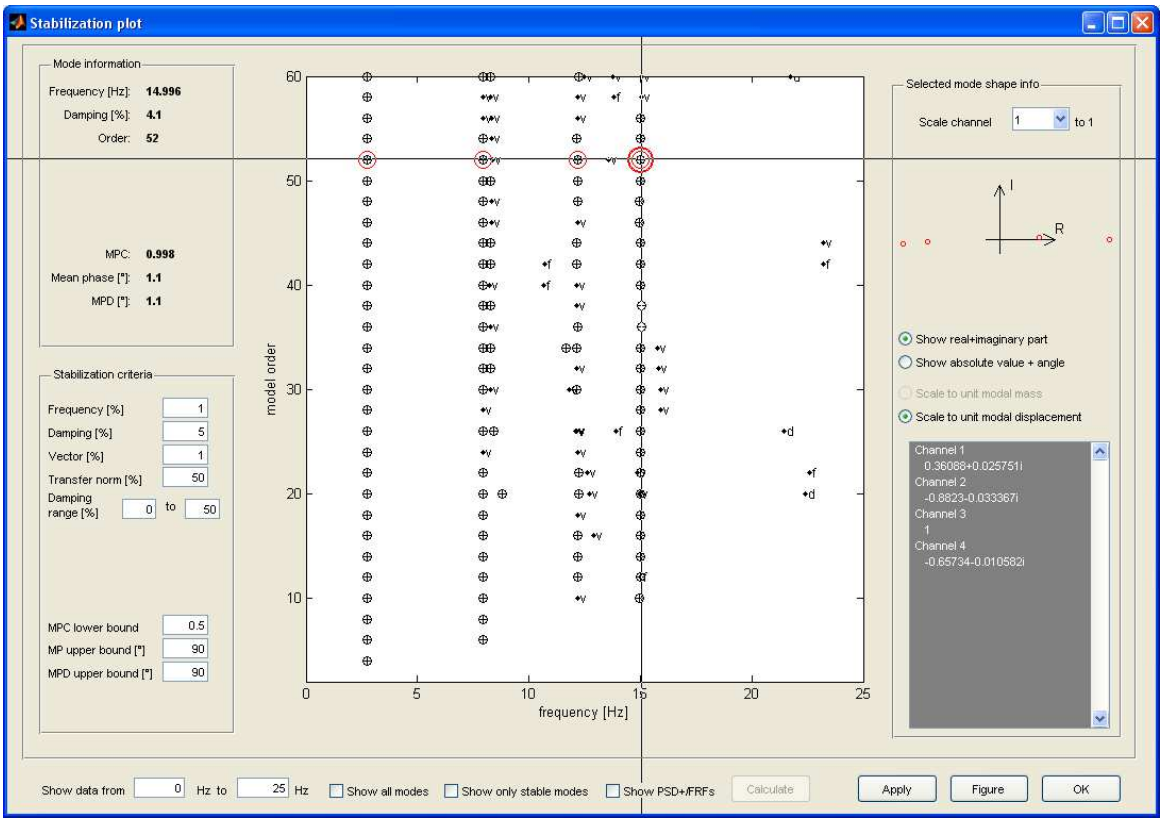


Figure 4.24: Tutorial 1: Stabilization diagram GUI.

42. Repeat the modal analysis with the `framedata_ssi_data_ref.mat` file. Now that you know how the PSD+ sum looks like, it is not necessary to plot it on the stabilization diagram. Choose for instance the four physical modes at a system order of 16 and save the mode information as

    `framedata_ssi_data_ref_modes.mat`

43. Select the `framedata_ssi_cov.mat` file in the MACEC main window and press again the MODAL ANALYSIS button. The stabilization diagram that is created is very clear and shows only physical modes, thanks to the variance calculation (fig. 4.25). If you move the cursor on a mode, the MODE INFORMATION panel shows, in addition to other information, the standard deviations of the eigenfrequency and damping ratio estimates as well as the maximum standard deviations of any of the real and imaginary mode shape components. Select the four different stable modes, for instance at a model order of 34, and save the mode information as

    `framedata_ssi_cov_modes.mat`

44. For the last modal analysis, select the `framedata_pLSCF.mat` file in the MACEC main window and press again the MODAL ANALYSIS button. The stabilization diagram looks very clear as the pLSCF method forces mathematical modes, that arise due to an over-estimation of the model order, to have negative damping. You can convince yourself of this property by setting the following values for the stabilization criteria:

    - Set the differences in eigenfrequency, damping ratio, mode shape and modal transfer norm between two consecutive model orders to 100%, 100%, 100%, and 5000%, respectively.

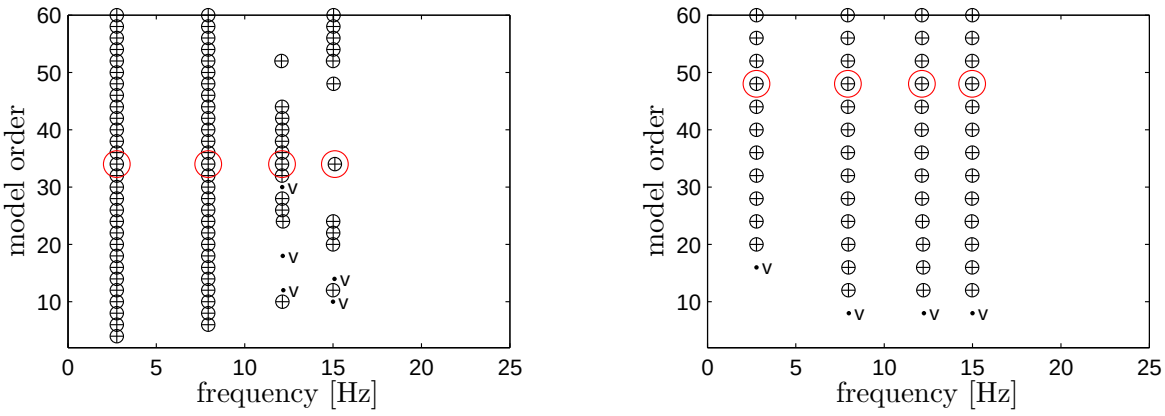


Figure 4.25: Tutorial 1: Stabilization diagram for the SSI-cov (left) and pLSCF (right) algorithms. ⊕ denotes a stable mode, and ·v, ·d and ·f a mode which satisfies all stabilization criteria except for the damping, mode shape, and damping and mode shape differences, respectively.

- Leave the damping range and the number of highest modal transfer norms unaltered.
- Change the modal phase collinearity lower bound to 0.
- Leave the mean phase and mean phase deviation upper bounds to their default value.

If you then press Apply, the stabilization diagram still shows only the four physical modes, although the only restrictive criterion is the damping range (fig. 4.25). But there is a price to pay: the linear least squares estimate for the RMFD model, obtained with pLSCF, is not statistically consistent when the transfer function has poles [9, pp. 199-200]. In practice, this means that the pLSCF estimate does not converge to the noiseless solution, even if a very large number of data points is available. However, we will see that this "bias" is, in general, quite small. After the physical modes have been selected, for instance at a model order of 48, save the mode information as

`framedata_pLSCF_modes.mat`

4.1.6 Results

45. The results of the Operational Modal Analysis can be viewed in graphical and numerical representation. Let's start with the graphical one. Select the `framedata_ssi_data_modes.mat` file in the MACEC main window and press the Plot mode shapes button in the Modal Analysis field. The Mode shape animation window now opens and the first mode is plotted. Since the dimensions of the structure under test are of the same order of magnitude as the modal displacements[11], change the Amplification field to *1* and press Enter. As the mode shapes are plane shapes in the XZ plane, push the view in the XZ plane button to change the viewpoint, as in fig. 4.26.

46. You can also play an animation of the mode shape by clicking the Play button in the Animation section. To save the mode shape as a figure, press the Figure button. To save the mode shape

[11] since output-only measurements were processed, the modal displacements have been automatically scaled to unit modal displacement.

animation as an `*.avi` file, press the MOVIE button. You can switch between the different modes using the > and < buttons or by selecting from the list in the MODE field. The mode shapes should look like in fig. 4.27.

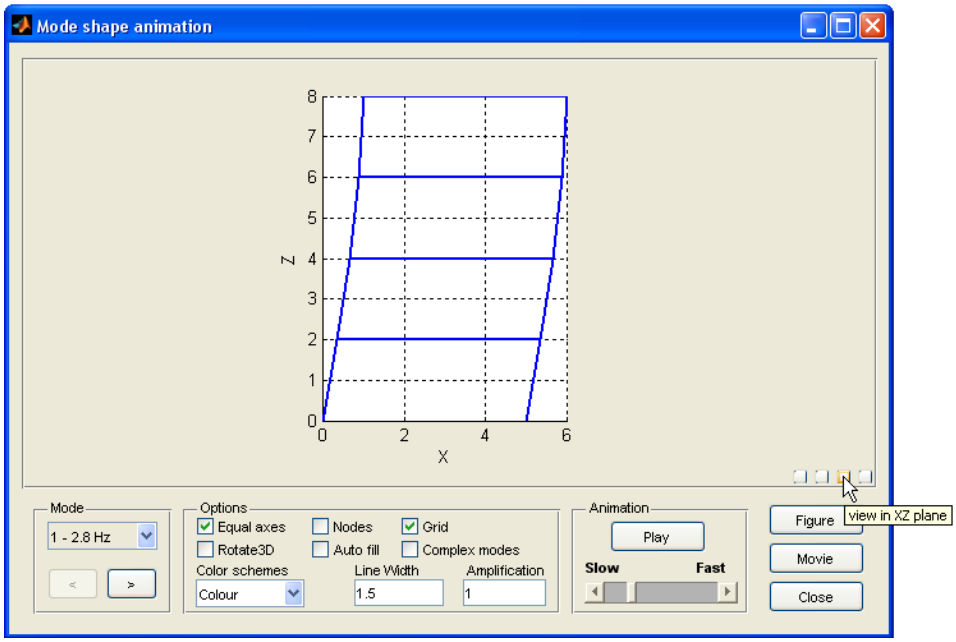


Figure 4.26: Tutorial 1: Plotting the first mode shape.

41. If you want to have a numerical presentation of the identified modal parameters, you can just select the appropriate file in the MACEC main window and press the LIST MODES button. The text editor opens and all modal information is plotted. Using this information, tables 4.1, 4.2 and 4.3 have been constructed.

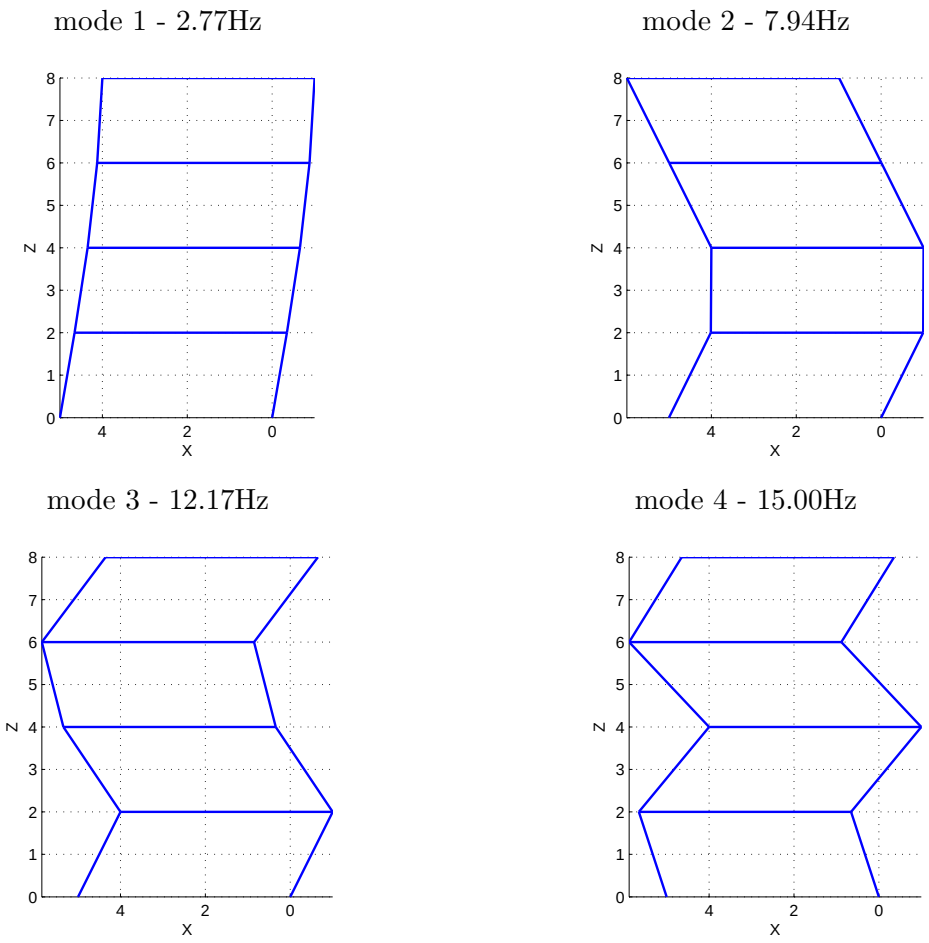


Figure 4.27: Tutorial 1: The four modes identified with SSI-data: eigenfrequencies and mode shapes.

4.1.7 Conclusions

Table 4.1 shows the true eigenfrequencies and the corresponding estimates, determined with six different methods. Since for the SSI-cov method, the standard deviations have been estimated, the 2σ confidence intervals, which corresponds to the 95% confidence intervals in the Gaussian case, are shown as well. The nonparametric estimates, obtained with PP and FDD, are the least accurate eigenfrequency estimates, as expected. The values obtained with both methods are exactly the same, due to the limited frequency resolution.

The eigenfrequencies obtained with the different parametric methods correspond very well. They all fall inside the 2σ confidence bound for the SSI-cov estimates. The uncertainty of the eigenfrequency of the third and fourth modes, which were less well excited (as can for instance be seen from the Power Spectral Density function, fig. 4.10), is somewhat higher than for the other modes. This can be seen both from comparing the confidence intervals of the SSI-cov estimates as well as from comparing the differences between the different methods.

mode	exact	PP	FDD	SSI-data	SSI-data/ref	SSI-cov	pLSCF
1	2.764	2.734	2.734	2.766	2.766	2.766 ± 0.014	2.772
2	7.958	8.008	8.008	7.944	7.946	7.935 ± 0.052	7.943
3	12.192	12.305	12.305	12.167	12.135	12.117 ± 0.281	12.132
4	14.956	14.844	14.844	14.997	15.161	15.097 ± 0.493	14.978

Table 4.1: Tutorial 1: True eigenfrequencies and estimates, obtained from six different methods, all in Hz. For the SSI-cov method, the values are shown together with their 2σ (95%) confidence interval.

The damping ratios, listed in table 4.2 correspond less well, as could have been expected from a short data record of only 8192 samples. This is also reflected in the high relative uncertainty of the damping ratio estimates of the SSI-cov method. In particular, although estimates of the damping ratio of the first mode agree much better than for the other modes, the SSI-cov uncertainty bound warns us that the accuracy is not better, which is confirmed by comparison with the exact value.

mode	exact	SSI-data	SSI-data/ref	SSI-cov	pLSCF
1	0.87	0.64	0.64	0.61 ± 0.37	0.68
2	2.50	1.89	2.38	2.26 ± 0.69	1.99
3	3.83	3.97	3.82	3.86 ± 2.55	2.65
4	4.70	4.15	3.60	3.97 ± 3.05	3.11

Table 4.2: Tutorial 1: True damping ratios and estimates, obtained from six different methods, all in % of critical. For the SSI-cov method, the values are shown together with their 2σ (95%) confidence interval.

MACEC offers additional useful validation information. When listing the identified modal parameters using the LIST MODES button in the MACEC main window, the standard deviations of the mode shape components are listed when available, as well as the modal phase collinearity (MPC), mean phase (MP), and mean phase deviation (MPD) values. When real normal modes are expected, as is the case for the proportionally damped structure considered here, the MPC, MP, and MPD values

should equal 1, 0° and 0°, respectively. Table 4.3 compares the MPD values for all methods. All values are very close to 0°, the limit value for real normal modes.

mode	PP	FDD	SSI-data	SSI-data/ref	SSI-cov	pLSCF
1	0.1	0	0.1	0.1	0.0	0.4
2	1.9	0	0.3	0.5	0.5	1.8
3	0.9	0	0.3	2.6	0.7	0.9
4	1.7	0	1.1	3.4	1.4	0.8

Table 4.3: Tutorial 1: Mean phase deviation (MPD) in degrees for the mode shapes, obtained from six different methods.

4.2 The B15 bridge

4.2.1 Introduction

In this tutorial, experimental data obtained on a large civil engineering structure are analyzed. The objective is to extract the modal characteristics from operational data that contain forced excitation, using the EMA and OMAX methods that are available in MACEC. The structure under test is the B15 bridge, which crosses the E19 highway that connects Brussels and Antwerp (Belgium). The bridge is located between the villages of Peutie and Perk (fig. 4.28).

Figure 4.28: Tutorial 2: View on the B15 bridge.

A schematic side view of the bridge and the deck cross section are shown in fig. 4.29. The B15 bridge, built in 1971, is a three-span box-girder bridge with an overall length of 124.6m ($35.8 + 53.0 + 35.8$m). The box-girder is 9.4m wide and varies in height between 1.0 and 2.5m. Including the two traffic lanes, it reaches a width of 13.0m.

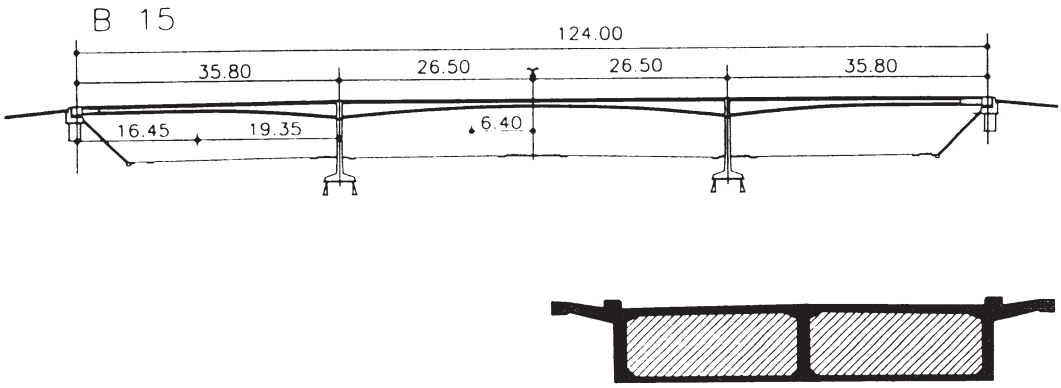


Figure 4.29: Tutorial 2: B15 bridge: side view and cross section.

An important feature of the B15 bridge is that it is skew-symmetric. Consequently, it can be expected

that there is no clear separation between bending and torsion modes, but that the vertical modes are a combination of bending and torsion [12]. The skew angle of the bridge is 43^o.

4.2.2 Vibration measurements

There are three types of dynamic vibration tests [11]: forced, ambient and combined. In forced vibration testing, it is assumed that all forces that are applied to the structure are measured. While this method yields very nice results for relatively small mechanical devices tested in laboratory conditions, it does not for large bridges because the so-called ambient forces like wind or traffic loads can not be measured and not be excluded, and large, impractical artificial excitation devices are needed to overcome this problem. Ambient vibration testing on the other hand is very well suited for bridges because in this type of testing, only responses (most often accelerations, sometimes also displacements, velocities or strains) at certain points of the structure need to be measured, not the forces. Finally, combined vibration testing consists of ambient vibration testing, but on top of the ambient excitation, also a forced excitation is applied in order to excite the structure over a broader frequency band and/or in order to have scaled mode shapes. The crucial difference with forced vibration testing is that the ambient loads are not considered as unwanted noise, but as a useful part of the excitation. Consequently, the amplitude of the artificial forces can be small compared to the amplitude of the ambient forces [13]. Combined vibration testing has raised interest only recently, because it requires special system identification methods. One of these methods is the CSI/ref method [12], which is incorporated into the MACEC software.

On the B15 bridge, both an ambient and a combined vibration test were performed. For the combined test, a falling weight (mass: 120kg, drop height: 1m) was used as an artificial, measured force, and the bridge was *not* closed to the road traffic. The system could be installed on the sidewalk, thus minimizing any disruption of traffic (fig. 4.30).

Figure 4.30: Tutorial 2: B15 bridge: drop weight setup.

The response of the bridge at selected points was measured only in the vertical direction using accelerometers (types: PCB type 393A, 393A03, 393C and Schaevitz). A spectrum analyser (type: Ono-Sokki) was used to control the measurement on site. Fig. 4.31 shows the experimental setup. In order not to disrupt the traffic on the bridge, the impact weight and the accelerometers were placed on the sidewalk and bicycle lane. The vertical accelerations at a total of 86 points (43 points per side) were measured (fig. 4.31). For the combined test, the falling weight was placed at node 25.

At the Antwerp side of the bridge, the vibration measurements were divided into 10 setups, each

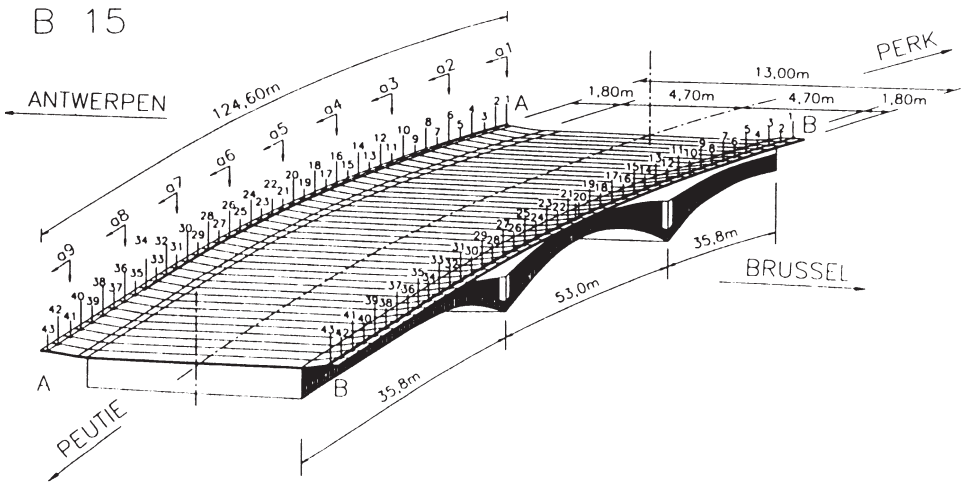


Figure 4.31: Tutorial 2: B15 bridge: measurement setup.

consisting of 11 Degrees Of Freedom (DOFs) measurements. Table 4.4 indicates which DOFs were measured in which setup. The force was of course only measured in the combined test.

setup	DOFs measured
1	$25Az$, $1Az$, $6Az$, $11Az$, $16Az$, $21Az$, $26Az$, $31Az$, $36Az$, $41Az$, $7Bz$, $25Afz$
2	$25Az$, $2Az$, $7Az$, $12Az$, $17Az$, $22Az$, $27Az$, $32Az$, $37Az$, $42Az$, $7Bz$, $25Afz$
3	$25Az$, $3Az$, $8Az$, $13Az$, $18Az$, $23Az$, $28Az$, $33Az$, $38Az$, $43Az$, $7Bz$, $25Afz$
4	$25Az$, $4Az$, $9Az$, $14Az$, $19Az$, $24Az$, $29Az$, $34Az$, $39Az$, $43Az$, $7Bz$, $25Afz$
5	$25Az$, $5Az$, $10Az$, $15Az$, $20Az$, $25Az$, $30Az$, $35Az$, $40Az$, $43Az$, $7Bz$, $25Afz$
6	$25Az$, $1Bz$, $6Bz$, $11Bz$, $16Bz$, $21Bz$, $26Bz$, $31Bz$, $36Bz$, $41Bz$, $7Bz$, $25Afz$
7	$25Az$, $2Bz$, $7Bz$, $12Bz$, $17Bz$, $22Bz$, $27Bz$, $32Bz$, $37Bz$, $42Bz$, $7Bz$, $25Afz$
8	$25Az$, $3Bz$, $8Bz$, $13Bz$, $18Bz$, $23Bz$, $28Bz$, $33Bz$, $38Bz$, $43Bz$, $7Bz$, $25Afz$
9	$25Az$, $4Bz$, $9Bz$, $14Bz$, $19Bz$, $24Bz$, $29Bz$, $34Bz$, $39Bz$, $43Bz$, $7Bz$, $25Afz$
10	$25Az$, $5Bz$, $10Bz$, $15Bz$, $20Bz$, $25Bz$, $30Bz$, $35Bz$, $40Bz$, $43Bz$, $7Bz$, $25Afz$

Table 4.4: Tutorial 2: B15 bridge: measurement setups and corresponding DOFs.

For the ambient vibration sessions, the ambient excitation was provided by the traffic underneath and on top of the bridge. During peak hours, the B15 bridge has intensive traffic making this vibration source the main cause of excitation. The total acquisition time for one output-only setup measurement was approximately 5 minutes at a sampling rate of 200Hz.

For the combined vibration measurements in which also the impact force was used and measured, the sampling frequency was chosen to be as high as 500Hz to capture the short-time transient signals of the impact in full detail. The measurement duration was about 12s.

4.2.3 Building the geometry

1. The first thing to do would be to construct the grid of measurement nodes, and to connect them into a surface model of the structure for visualization. Because the GEOMETRY part of the MACEC main window has already been treated in detail in the previous tutorial (section 4.1.2), the construction of the grid and the surface files is not explained in detail here, but the files have been prepared on beforehand. You can simply load the `b15_grid.asc` file from the `..\spice\tutorials\tutorial2` directory using the SELECT GRID FILE button in the GEOMETRY section of the MACEC main window (fig. 4.32). If you want, you can have a look at the defined measurement nodes by using the EDIT GRID FILE button which is located next to the SELECT GRID FILE button.

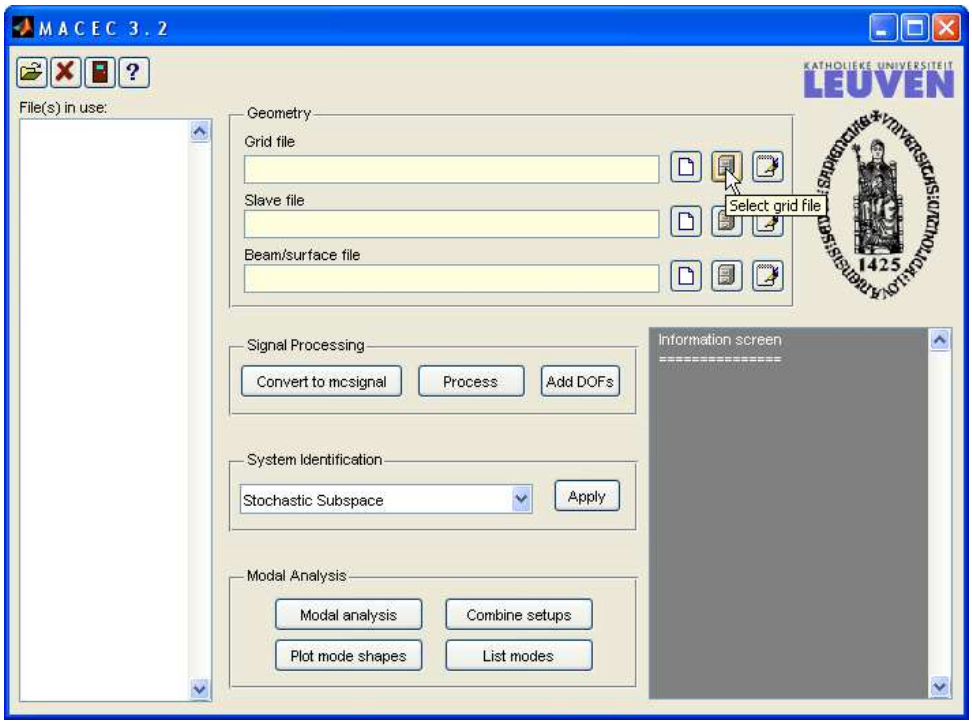


Figure 4.32: Tutorial 2: Loading a GRID file.

2. In the same way, select the `b15_slave.asc` file from the `..\spice\tutorials\tutorial2` directory using the SELECT SLAVE FILE button in the GEOMETRY section of the MACEC main window. Afterwards, select the `b15_surface.asc` file from the `..\spice\tutorials\tutorial2` directory using the SELECT BEAM OR SURFACE FILE button in the GEOMETRY section of the MACEC main window.

4.2.4 Processing the measured signals

The geometry of the B15 bridge has now been loaded, so the signal processing part can start.

3. Select the files with the simulated measurement data by pushing the SELECT NEW DATA... button in the MACEC main window. Choose the files

   ```
   VALA1-1.F32
   VALA2-1.F32
   VALA3-1.F32
   VALA4-1.F32
   VALA5-1.F32
   VALB1-1.F32
   VALB2-1.F32
   VALB3-1.F32
   VALB4-1.F32
   VALB5-1.F32
   ```

   in the `..\spice\tutorials\tutorial2\measurements` directory.

4. The names of these files appear in the FILE(S) IN USE list in the MACEC main window. Select all files and press the CONVERT TO MCSIGNAL button in the SIGNAL PROCESSING section to open the CONVERSION OF THE MEASURED DATA window. In the previous tutorial, it was indicated how you can convert the measurement data to an MCSIGNAL object using this window, so this is not explained in detail here, but the sensitivities, amplification factors, conversion factors, labels and units have been saved on beforehand. Therefore, press the LOAD button (fig. 4.33) and select the file

   ```
   sens-F.mat
   ```

   in the `..\spice\tutorials\tutorial2` directory. All settings are now automatically adjusted, and you can make the conversion to the MCSIGNAL object by pressing the OK button. Save the first MCSIGNAL object as

   ```
   VALA1-1_conv.mat
   ```

   Save the other MCSIGNAL objects accordingly (save the MCSIGNAL object created from the file `VALA2-1.F32` as `VALA2-1_conv.mat`, etc.). Note that it was possible to convert the data for all 10 setups at the same time because in each setup, the same physical quantities are measured using the same hardware and the same amplification factors.

5. In the MACEC main window, select the `VALA1-1_conv.mat` file in the FILE(S) IN USE list and press the PROCESS button in the SIGNAL PROCESSING section to open the PREPROCESS MCSIGNAL OBJECT window. Here you can have a look at the raw measurement data and their frequency content, or, if you change the TIME-FREQUENCY setting in the VISUALIZATION section to AUTOCORRELATION-PSD, at the autocorrelation function and the Power Spectral Density (PSD) (fig. 4.34). Also note that the axis labels are automatically adjusted in the right way, thanks to the definition of the signal's units during the conversion to an MCSIGNAL object.

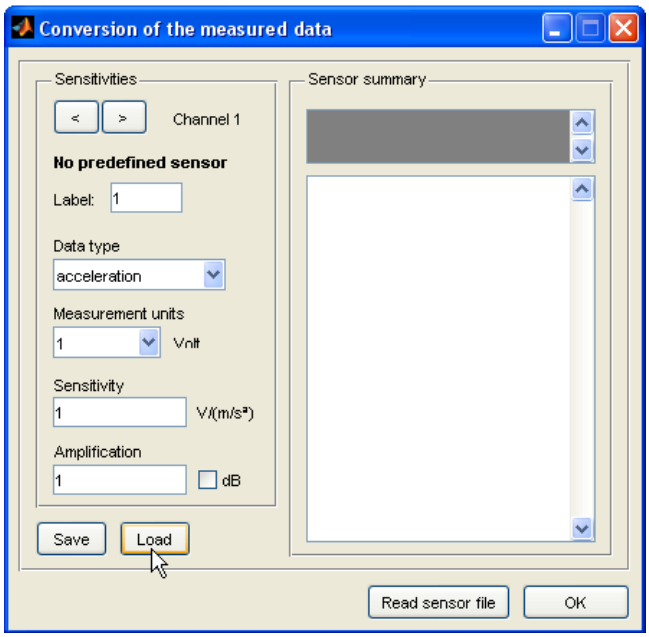


Figure 4.33: Tutorial 2: Loading a saved file with sensitivities, amplification factors, conversion factors, labels and units.

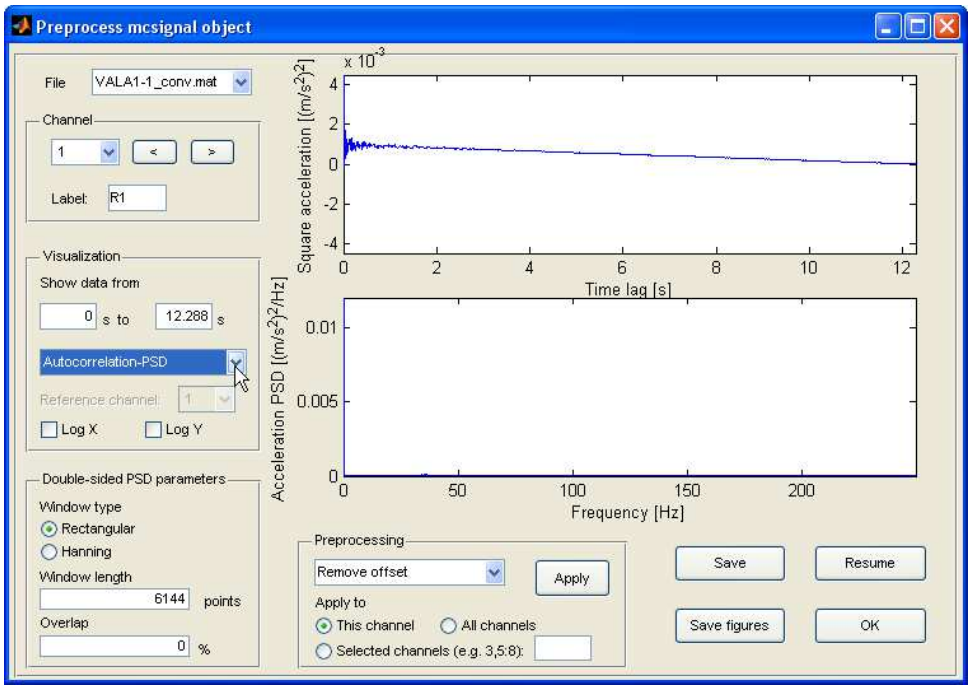


Figure 4.34: Tutorial 2: Plotting the autocorrelation and power spectral density of a channel's signal.

6. When you have a look at the autocorrelation signal of the first channel (fig. 4.34), you notice that there is an offset on the signals that needs to be removed first. Therefore, select Remove offset in the Preprocessing section, select All channels and press Apply (fig. 4.35). When switching through the channels, you now indeed observe, both from the autocorrelation and the PSD plots, that the constant trend has been removed.

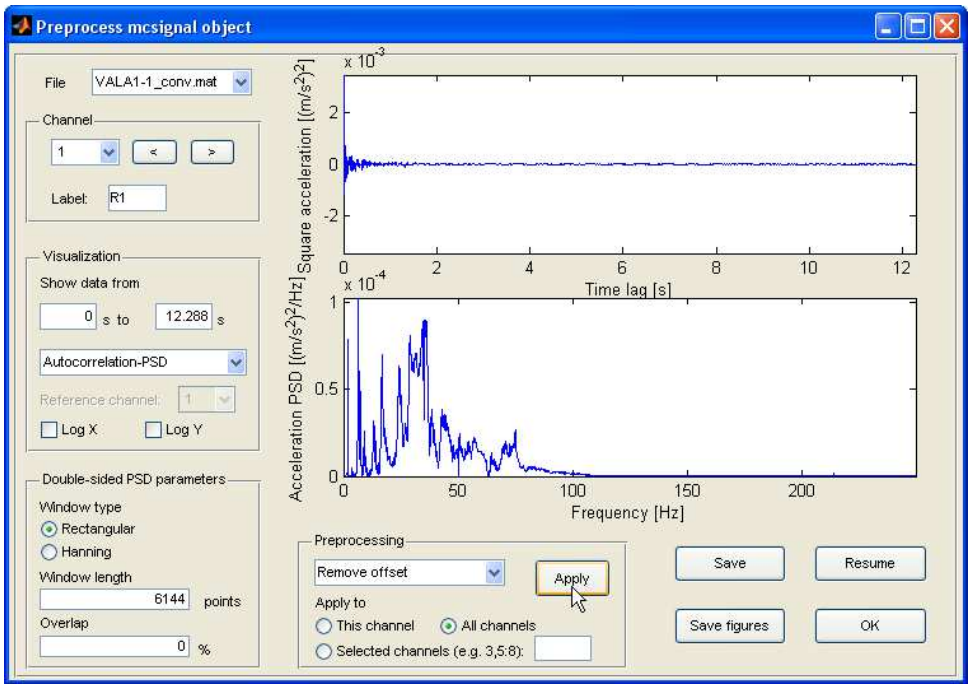


Figure 4.35: Tutorial 2: Removing the offset from the signals.

7. Now let's have a look at the PSD of both reference accelerations (channels 1 and 11) and the force signal (channel 12). You can easily notice that the main part of the signal's energy lies between 0 and $100 - 150$Hz. However, in bridge engineering the frequency range of interest is between 0 and 20Hz. In order to perform a data reduction and to facilitate the System Identification, let's first apply digital filtering to the signals and then re-sample them, so that we have now a Nyquist frequency of 25Hz instead of 250Hz. Hereto, select DECIMATE[12] in the PREPROCESSING section and press APPLY. MACEC now asks for the decimation factor, an integer by which the number of samples will be divided. Choose *10* and press OK (fig. 4.36).

8. After the decimation, press OK and save the processed MCSIGNAL object as

   `VALA1-1_proc.mat`

   Repeat the preprocessing steps for the other setups. In the fifth setup, the sixth channel should be removed, in the sixth setup the eighth channel, and in the seventh setup the third channel. In order to do so, select DELETE CHANNEL in the PREPROCESSING part of the PREPROCESS MCSIGNAL OBJECT window, select the right channel, press APPLY and confirm your choice (fig. 4.37). Note that you can process several MCSIGNAL objects at the same time by selecting more than one object in the MACEC main window; however, not all signal processing operation are allowed then on the individual MCSIGNAL objects.

9. To end the preprocessing, the measured DOFs have to be defined for each channel. Select the `VALA1-1_proc.mat` file in the FILE(S) IN USE section of the MACEC main window and press the

[12] DECIMATE is a standard function from the Signal Processing Toolbox of MATLAB. For more information about this function, consult the MATLAB help or the MATLAB documentation.

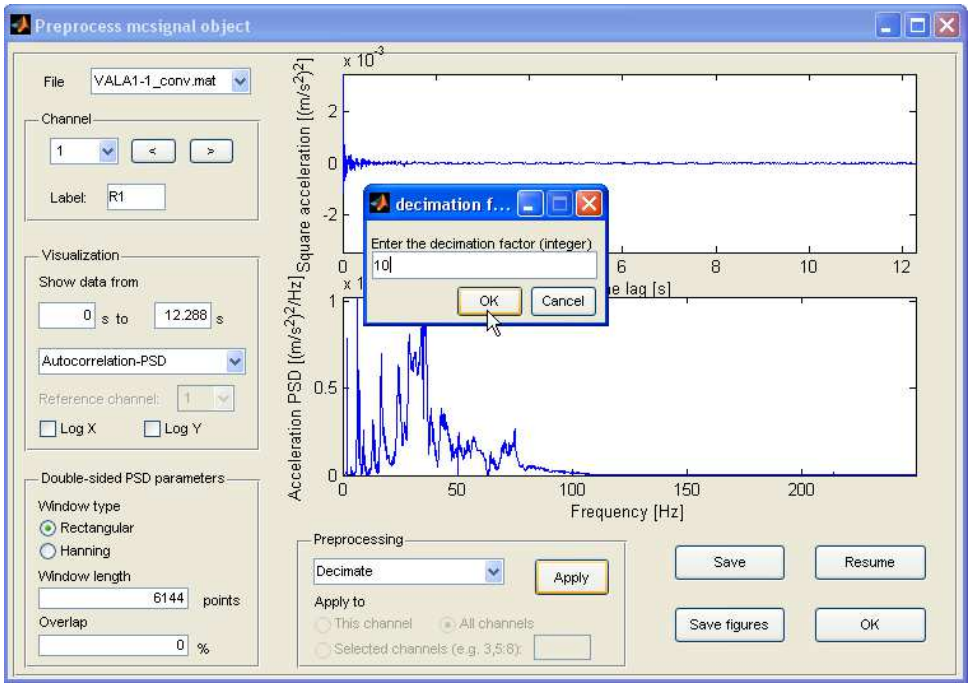


Figure 4.36: Tutorial 2: Decimation with a factor 10.

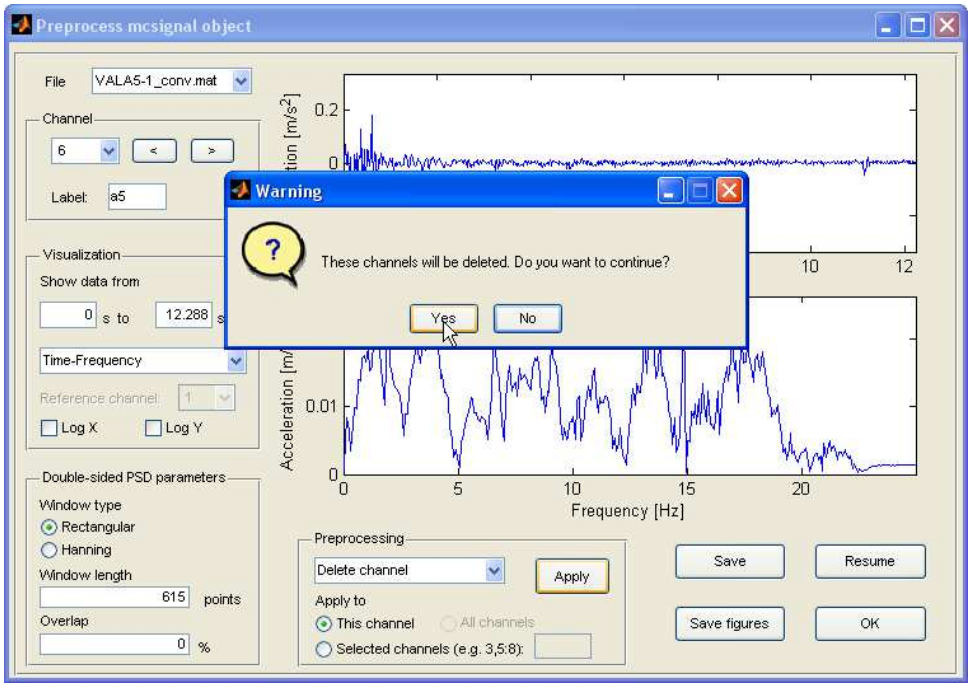


Figure 4.37: Tutorial 2: Setup 5: deleting channel 6.

ADD DOFS button in the SIGNAL PROCESSING section. Because the definition of the DOFs has already been treated in the first tutorial (section 4.1.3), they have been prepared on beforehand in this case. In the ADD CHANNEL SPECIFICATION window, press the LOAD button and choose

`b15_dofs_F1.mat`

All DOF information is now adjusted. Press OK and then YES to save the DOF information. Repeat this for the other setups.

4.2.5 System identification

In MACEC, classical experimental modal analysis (EMA) is possible with nonparametric frequency response function (FRF) estimation using the classical H_1 estimator [5], and with the deterministic pLSCF method, which is a parametric method that starts from a nonparametric FRF description. In EMA, the influence of the unmeasured ambient forces is considered as disturbing noise; it is removed in the nonparametric FRF estimation.

Combined vibration testing in MACEC is possible with the data-driven reference-based Combined deterministic-stochastic Subspace Identification (CSI/ref) method. It has the advantage that both the measured drop weight force and the unmeasured ambient forces are accounted for.

Nonparametric FRF estimation

10. Select the `VALA1-1_proc.mat` file in the FILE(S) IN USE section of the MACEC main window, select NONPARAMETRIC (FOR PEAK PICKING OR CMIF/FDD) in the SYSTEM IDENTIFICATION field and press APPLY to open the POLY-REFERENCE LEAST SQUARES COMPLEX FREQUENCY DOMAIN (PLSCF) IDENTIFICATION window. You can see that MACEC immediately suggests, based on the type of data, which channels will be considered as inputs and which channels will be outputs, and which ANALYSIS TYPE is needed. This suggestion is correct, so you do not have to change it (fig. 4.38). Press APPLY in the FRF ESTIMATION field to start the estimation of a non-parametric FRF using the H_1 estimator [5].

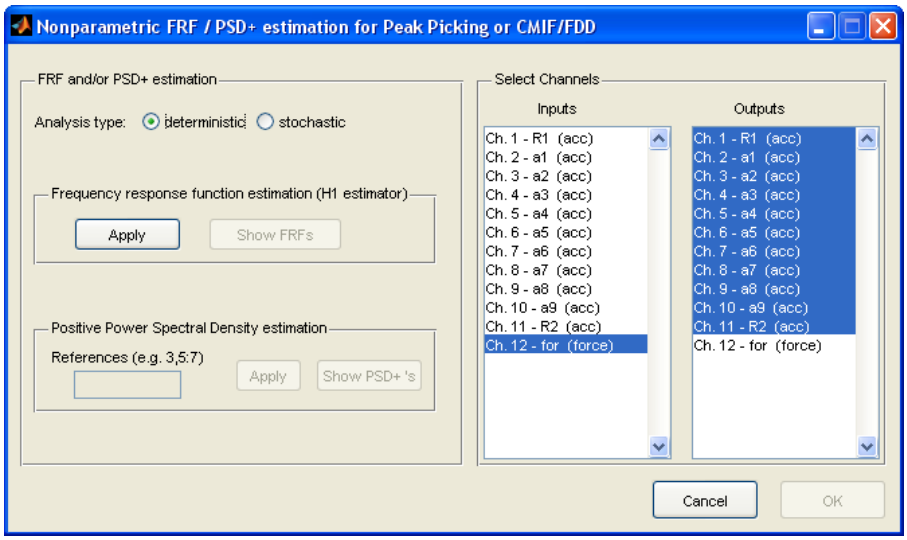


Figure 4.38: Tutorial 2: Deterministic nonparametric system identification.

11. In the FRF ESTIMATION (H1 METHOD) window, you have to specify a number of data blocks first. As the total number of samples available is small and the minimal number of 2 blocks is automatically suggested by MACEC, you don't have to change anything. Just press the CALCULATE AND SHOW ESTIMATES button to start the H1 estimation. If this is finished, you

can see from the Resolution info panel that the frequency resolution is quite course (0.1626Hz) (fig. 4.39).

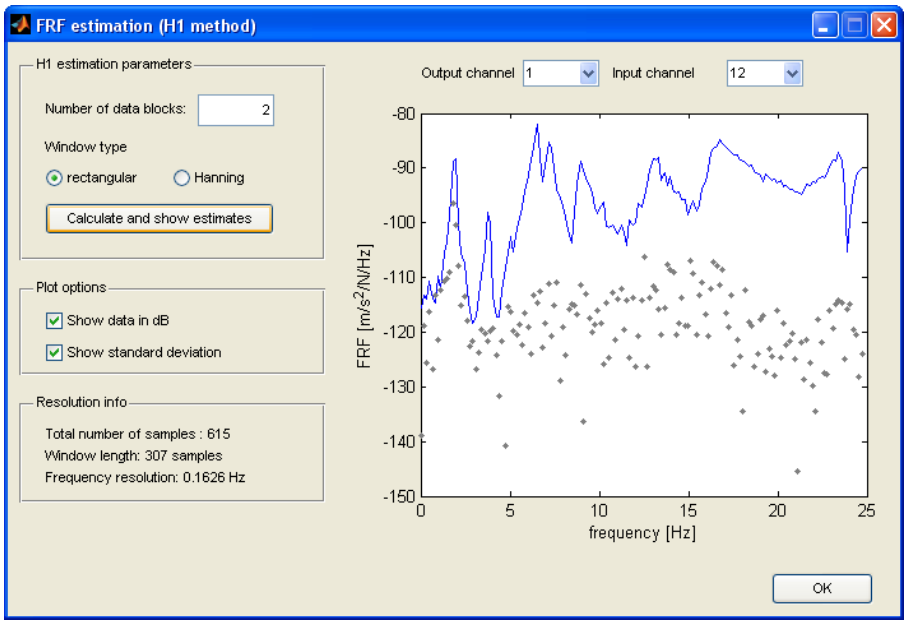


Figure 4.39: Tutorial 2: H_1 FRF estimation for pLSCF identification.

12. After pressing OK, MACEC returns to the Nonparametric FRF / PSD+ estimation for Peak Picking or CMIF/FDD window. In this window, press OK and save the models in the following `*.mat` file:

    `VALA1-1_nonpar`

13. Repeat the identification step for the other setups. You can have a look at the `logfile_cmif.m` file in the `tutorial2` section in order to have an idea on how this can be performed in a batch run, i.e. without using the GUI.

deterministic pLSCF

14. Select the `VALA1-1_proc.mat` file in the File(s) in use section of the MACEC main window, select pLSCF in the System Identification field and press Apply to open the Poly-reference Least Squares Complex Frequency Domain (pLSCF) Identification window. You can see that MACEC immediately suggests, based on the type of data, which channels will be considered as inputs and which channels will be outputs, and which Analysis type is needed. This suggestion is correct, so you do not have to change it. Press Apply in the FRF estimation field (fig. 4.40) to start the estimation of a non-parametric FRF using the H_1 estimator [5]. Follow exactly the same steps as in paragraph 11 from the previous section.

15. After pressing OK, MACEC returns to the Poly-reference Least Squares Complex Frequency Domain (pLSCF) Identification window, where it is now possible to calculate the system matrices of a Right Matrix Fraction Description (RMFD) model using the pLSCF algorithm. Since the bandwidth of interest coincides with the available frequency band, don't change the default values in the Frequency range fields. In the Polynomial orders field, fill in

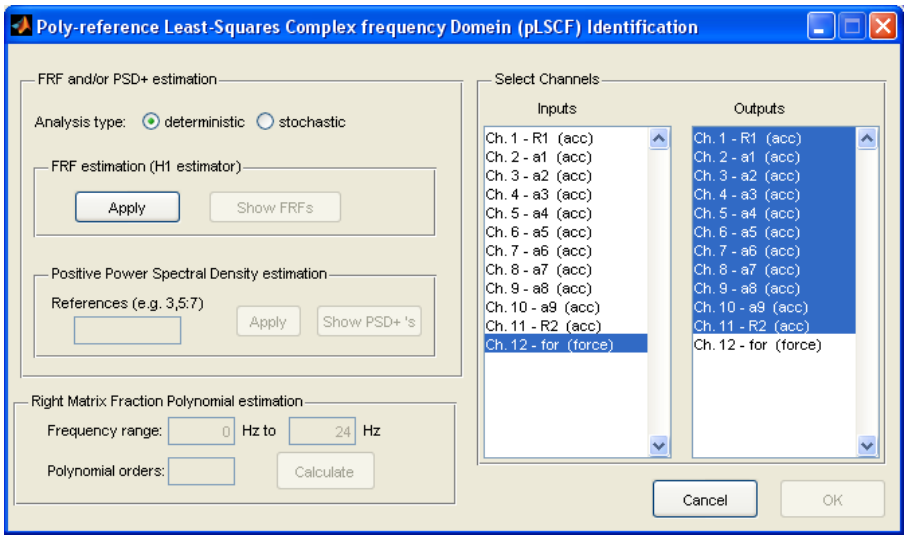


Figure 4.40: Tutorial 2: Deterministic pLSCF.

2:2:100

Since there's only one input, the polynomial order range equals the system order range. If the identification is ready, press OK and save the identified system matrices as

`VALA1-1_pLSCF.mat`

16. Repeat the identification step for the other setups. You can have a look at the `logfile_pLSCF.m` file in the `tutorial2` section in order to have an idea on how this can be performed in a batch run, i.e. without using the GUI.

CSI/ref

17. Select the `VALA1-1_proc.mat` file in the FILE(S) IN USE section of the MACEC main window, select COMBINED SUBSPACE in the SYSTEM IDENTIFICATION field and press APPLY to open the COMBINED DETERMINISTIC-STOCHASTIC SUBSPACE IDENTIFICATION window[13]. At the right, you can see that MACEC immediately suggests which channels have to be considered as inputs and which channels are outputs, based on the data type. In the ALGORITHM SELECTION field, mark the REFERENCE-BASED box and indicate that the channels with label R1 and R2 (for this setup, channels 1 and 11), the DOFs of which are common to every setup, are the reference channels (fig. 4.41).

18. Then, choose *40* as half the number of block rows i [14] and press the CALCULATE QR + SVD button to start the first part of the identification. If this is finished, choose

2:2:80

[13]The theory behind the CSI/ref algorithm is not explained in this manual. The interested reader is referred to [12] for more information.

[14]See [12] for a discussion on the relationship between i and the lowest frequency of interest.

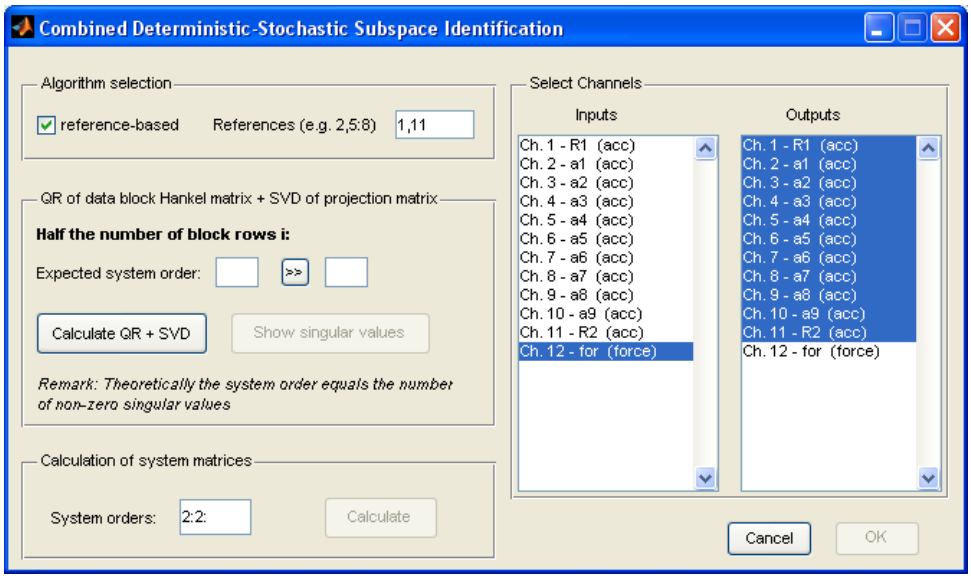


Figure 4.41: Tutorial 2: Reference-based Combined deterministic-stochastic Subspace Identification (CSI/ref).

as the model orders to be calculated and press CALCULATE. This calculation might take several minutes. If the system matrices have been calculated, press OK and save them as

`VALA1-1_CSIref.mat`

19. Repeat this identification step for the other setups.

4.2.6 Modal Analysis

With the Complex Mode Indication Function

20. In the MACEC main window, select `VALA1-1_nonpar.mat` and press the MODAL ANALYSIS button in the MODAL ANALYSIS field. The complex mode indication function (CMIF) [16] is computed and plotted (fig. 4.42). Since only one input, the drop weight, was used in the test, the CMIF consists of one singular value only, and as such properly identifying closely spaced modes is not possible. As noted before, the short measurement duration results in a very coarse frequency resolution. Since for this bridge, it is expected that the mass-normalized mode shapes are purely real (no double modes, no localized dampers), the selected modes should have a high modal phase collinearity (MPC) value and a low mean phase deviation (MPD) value. By inspecting the CMIF by moving the cursor over it, it can be verified that only two peaks have an MPC value larger than 0.9. Select the peaks as indicated in fig. 4.42; the other peaks will not be present in all setups or will yield very poor modal parameter estimates in some setups.

21. After the poles have been selected, save them as

`VALA1-1_CMIF_modes.mat`

Repeat the mode selection step for all 9 other setups.

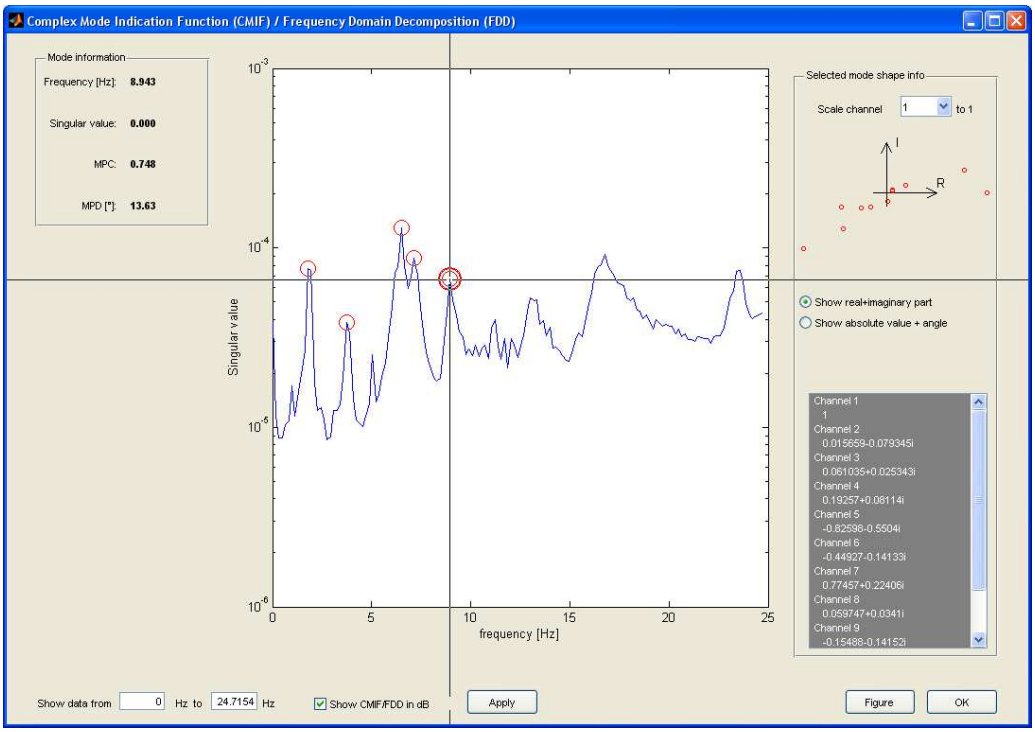


Figure 4.42: Tutorial 2: CMIF function for the first measurement of the first setup. The selected modes are indicated with red circles.

22. If the modes of all 10 setups have been selected, we can make use of MACEC's possibility to combine modal information obtained from different setups into one single mode. Hereto, select `VALA1-1_cmif_modes.mat` and the mode information for the nine other setups in the FILE(S) IN USE section of the MACEC main window and press the COMBINE SETUPS button in the MODAL ANALYSIS field. Save the resulting modes as

    `VAL-1_cmif_modes.mat`

23. Now select this file in the FILE(S) IN USE section of the MACEC main window and press the PLOT MODE SHAPES button in the MODAL ANALYSIS section to get a look at the different mode shapes. You notice that some of them look quite well, why others are more noisy due to the small amount of data that was available for the identification.

24. As each impact test was repeated 4 times, there are 4 measurements per setup. If you would repeat the complete modal identification procedure (signal processing, system identification and modal analysis) for the other tests and take the mean value of the modal information obtained for each test at a particular setup, you can expect that the results will be more accurate. Taking this mean value is not difficult: if two files with modal information containing exactly the same DOFs are selected and the COMBINE SETUPS button is pushed, the resulting modal data contain the mean values.

25. As the previous step is quite time-consuming and does not imply new functionalities of MACEC, you can just select

    `VAL_cmif_modes.mat`

which contains the results. If you plot these mode shapes, the result looks like in fig. 4.43. The quality of the mode shape of mode 3 is less than for the other modes. In all identified modes, bending is combined with torsion, due to the skewness of the bridge supports with respect to the bridge deck.

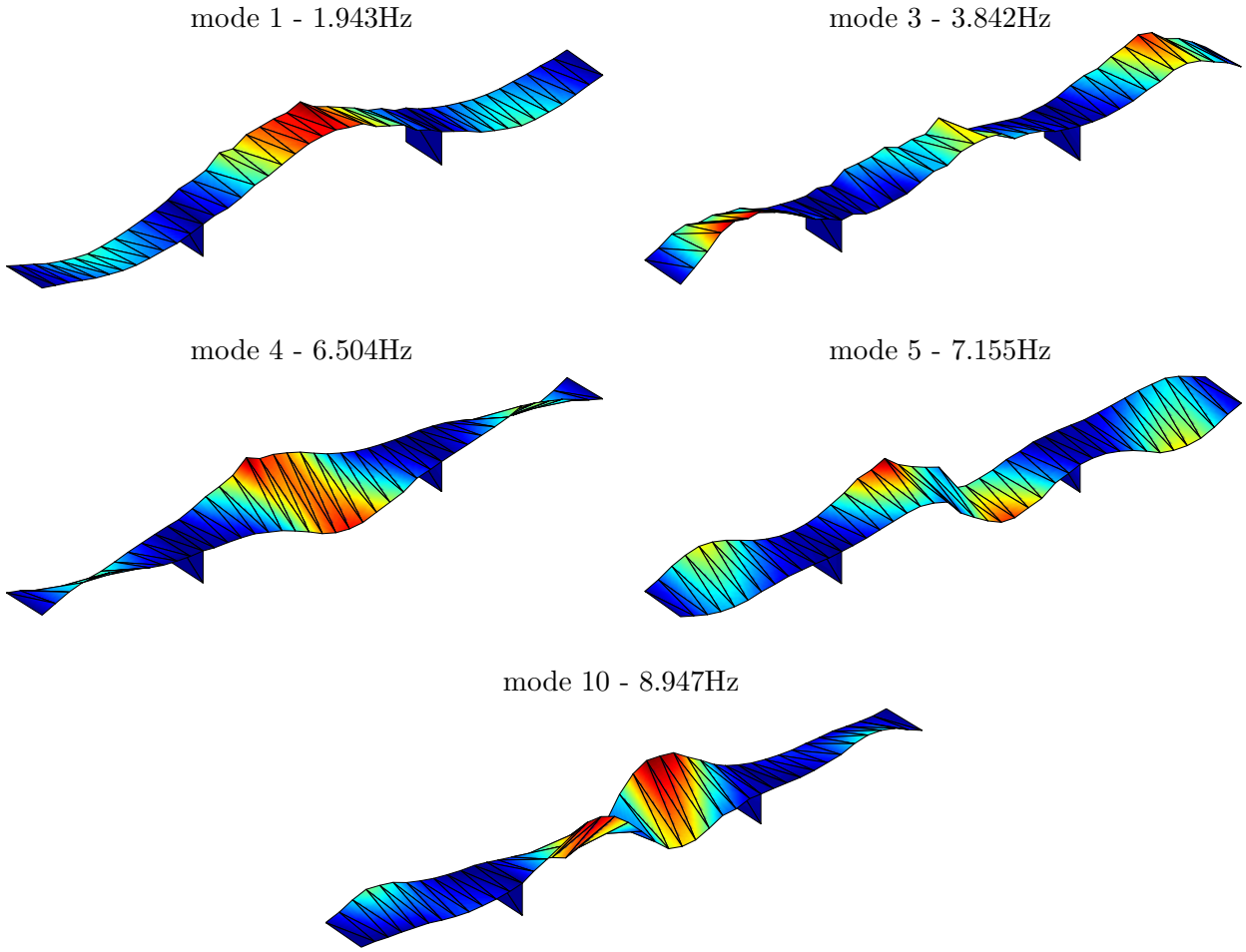


Figure 4.43: Tutorial 2: Eigenfrequencies and mode shapes obtained with CMIF.

After deterministic pLSCF

26. In the MACEC main window, select `VALA1-1_pLSCF.mat` and press the Modal analysis button in the Modal Analysis field. The stabilization diagram that is now constructed is very clear, so you can easily pick the nine stable modes at order 80. Note that, since it concerns a forced vibration test, the modes have been mass-normalized.

27. After the poles have been selected, save them as

    `VALA1-1_pLSCF_modes.mat`

    Repeat the pole selection step for all 9 other setups.

28. If the poles of all 10 setups have been selected, we can make use of MACEC's possibility to combine modal information obtained from different setups into one single mode. Hereto, select `VALA1-1_pLSCF_modes.mat` and the mode information for the nine other setups in the FILE(S) IN USE section of the MACEC main window and press the COMBINE SETUPS button in the MODAL ANALYSIS field. Save the resulting modes as

    `VAL-1_pLSCF_modes.mat`

29. You can have a look at the different mode shapes by selecting this file in the FILE(S) IN USE section of the MACEC main window and pressing the PLOT MODE SHAPES button in the MODAL ANALYSIS section. Note that with the pLSCF method, more modes could be identified than with CMIF.

30. As each impact test was repeated 4 times, there are 4 measurements per setup. If you would repeat the complete modal identification procedure (signal processing, system identification and modal analysis) for the other tests and take the mean value of the modal information obtained for each test at a particular setup, you can expect that the results will be more accurate. Taking this mean value is not difficult: if two files with modal information containing exactly the same DOFs are selected and the COMBINE SETUPS button is pushed, the resulting modal data contain the mean values.

31. As the previous step is quite time-consuming and does not imply new functionalities of MACEC, you can just select

    `VAL_pLSCF_modes.mat`

    which contains the results. If you plot these mode shapes, the result looks like in fig. 4.44. The quality of the mode shapes of modes 4 and 9 is less than for the other modes. In all identified modes, bending is combined with torsion, due to the skewness of the bridge supports with respect to the bridge deck.

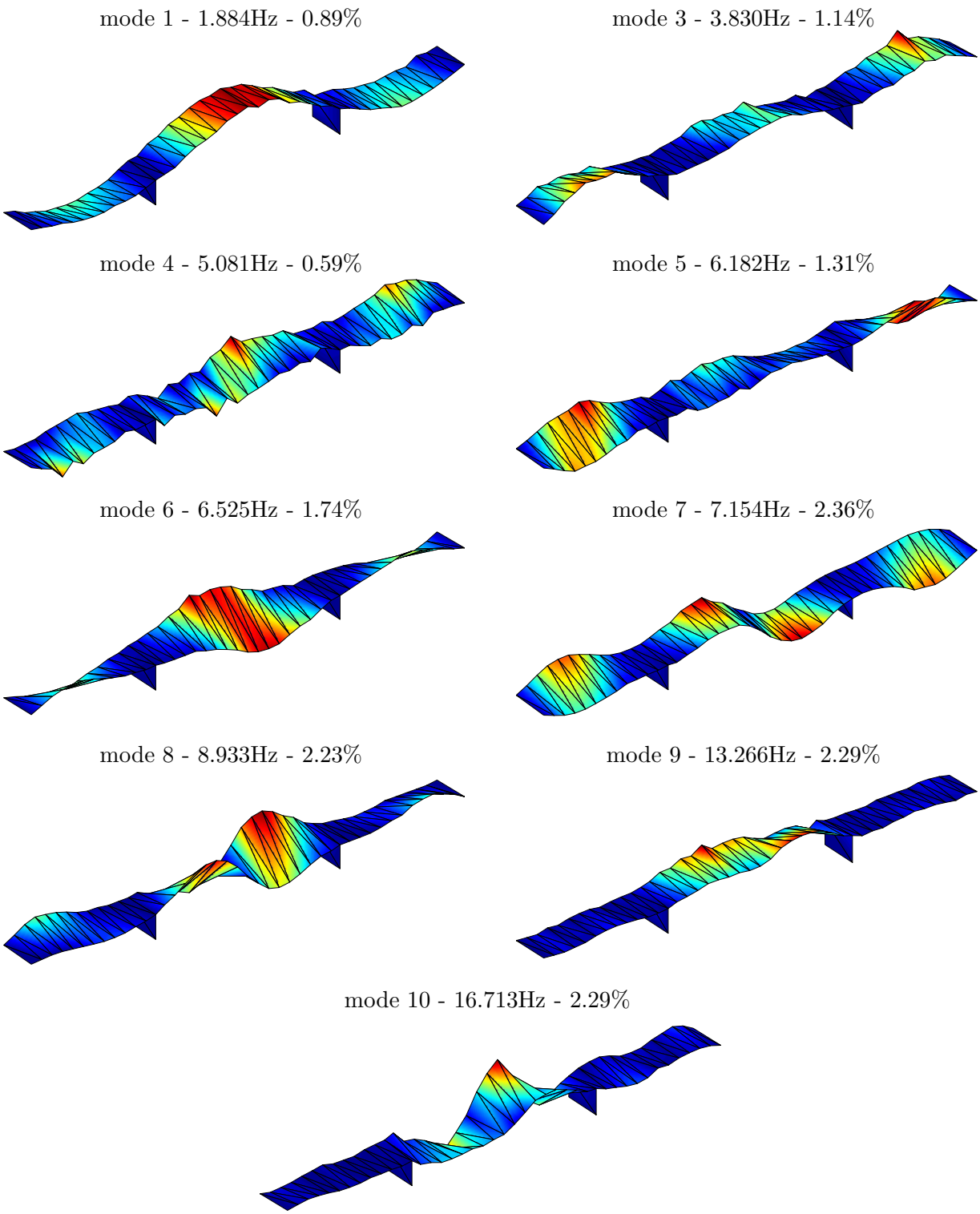


Figure 4.44: Tutorial 2: Eigenfrequencies, damping ratios and mode shapes obtained with deterministic pLSCF.

After CSI/ref

31. In the MACEC main window, select `VALA1-1_CSIref.mat` and press the MODAL ANALYSIS button in the MODAL ANALYSIS field. The stabilization diagram that is now constructed is clear, but looks incomplete, indicating that the default stabilization criteria are too strict. Therefore, relax the TRANSFER NORM [%] criterion to 75 % and press APPLY. The stabilization diagram now consists of clear vertical lines between 0 and 10Hz, but it is still hard to interpret between 10 and 25Hz (fig. 4.45). This can be explained from the sum of the FRFs, that is plotted on top of the stabilization diagram in fig. 4.45: above 10Hz, many modes are present, that are probably not well excited. The most important peaks in the FRF sum are situated around 13 and 16.5Hz, which corresponds to modes 9 and 10 from fig. 4.44. Although it is also possible to pick these modes from the stabilization diagram obtained after CSI/ref identification, it is a time-consuming task due to the unclear stabilization. Therefore, we confine ourselves here to the modes between 0 and 10Hz.

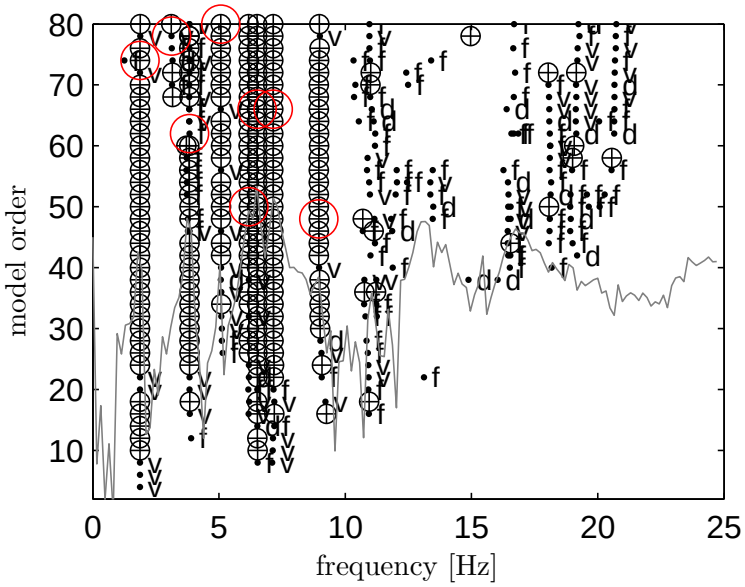


Figure 4.45: Tutorial 2: Stabilization diagram constructed from state-space systems identified with CSI/ref (first setup, first measurement, default stabilization criteria). ⊕ denotes a stable mode, ·v a stable frequency and mode shape, ·d a stable frequency and damping, and ·f a stable frequency. The selected modes are indicated with red circles.

32. In the frequency range between 0 and 10Hz, one can easily find the eight stable modes indicated in table 4.5. Note that, within a column of stable modes, the estimates with the highest MP or MPC values are selected. The mode at 3.113Hz could not be identified using deterministic pLSCF, simply because it was not present in the stabilization diagram. Another remarkable difference can be noticed at 3.825Hz: although at this frequency only one stable mode appeared in the pLSCF stabilization diagram, from the CSI/ref stabilization diagram we notice that in reality two very closely spaced modes are present around this frequency. For example: in fig. 4.45, at a model order of 60, we have a stable pole at 3.709Hz (damping ratio of 4.2%) and one at

f[Hz]	1.877	3.119	3.830	5.072	6.181	6.513	7.150	8.957
ξ[%]	1.4	3.0	1.1	1.3	1.6	1.8	2.3	2.7
model order	74	78	62	80	50	66	66	48

Table 4.5: Tutorial 2: Eigenfrequencies and damping ratios for setup 1, determined using CSI/ref.

3.825Hz (damping ratio of 1.3%). However, we advise not to select the pole at 3.709Hz because it will show up in most but not all setups.

33. After the poles have been selected, save them as

    `VALA1-1_CSIref_modes.mat`

    Repeat the pole selection step for all 9 other setups.

34. If the poles of all 10 setups have been selected, we can make use of MACEC's possibility to combine modal information obtained from different setups into one single mode. In order to do this, select `VALA1-1_CSIref_modes.mat` and the mode information for the nine other setups in the File(s) in use section of the MACEC main window and press the Combine setups button in the Modal Analysis field. Save the resulting modes as

    `VAL-1_CSIref_modes.mat`

35. Now select this file in the File(s) in use section of the MACEC main window and press the Plot mode shapes button in the Modal Analysis section to get a look at the different mode shapes. You notice that some of them look quite well, why others are more noisy due to the small amount of data that was available for the identification.

36. As each impact test was repeated 4 times, there are 4 measurements per setup. If you would repeat the complete modal identification procedure (signal processing, system identification and modal analysis) for the other tests and take the mean value of the modal information obtained for each test at a particular setup, you can expect that the results will be more accurate. Taking this mean value is not difficult: if two files with modal information containing exactly the same DOFs are selected and the Combine setups button is pushed, the resulting modal data contain the mean values.

37. As the previous step is quite time-consuming and does not imply new functionalities of MACEC, you can just select

    `VAL_CSIref_modes.mat`

    which contains the results. If you plot these mode shapes, the result looks like in fig. 4.46. The quality of the mode shape of mode 2 is less than for the other modes. This should not be a surprise, since this mode is only weakly excited and it could even not be detected with deterministic pLSCF. Modes 4 and 5 are clearly of better quality than the same mode identified with deterministic pLSCF (compare with fig. 4.44). In all identified modes, bending is combined with torsion, due to the skewness of the bridge supports with respect to the bridge deck.

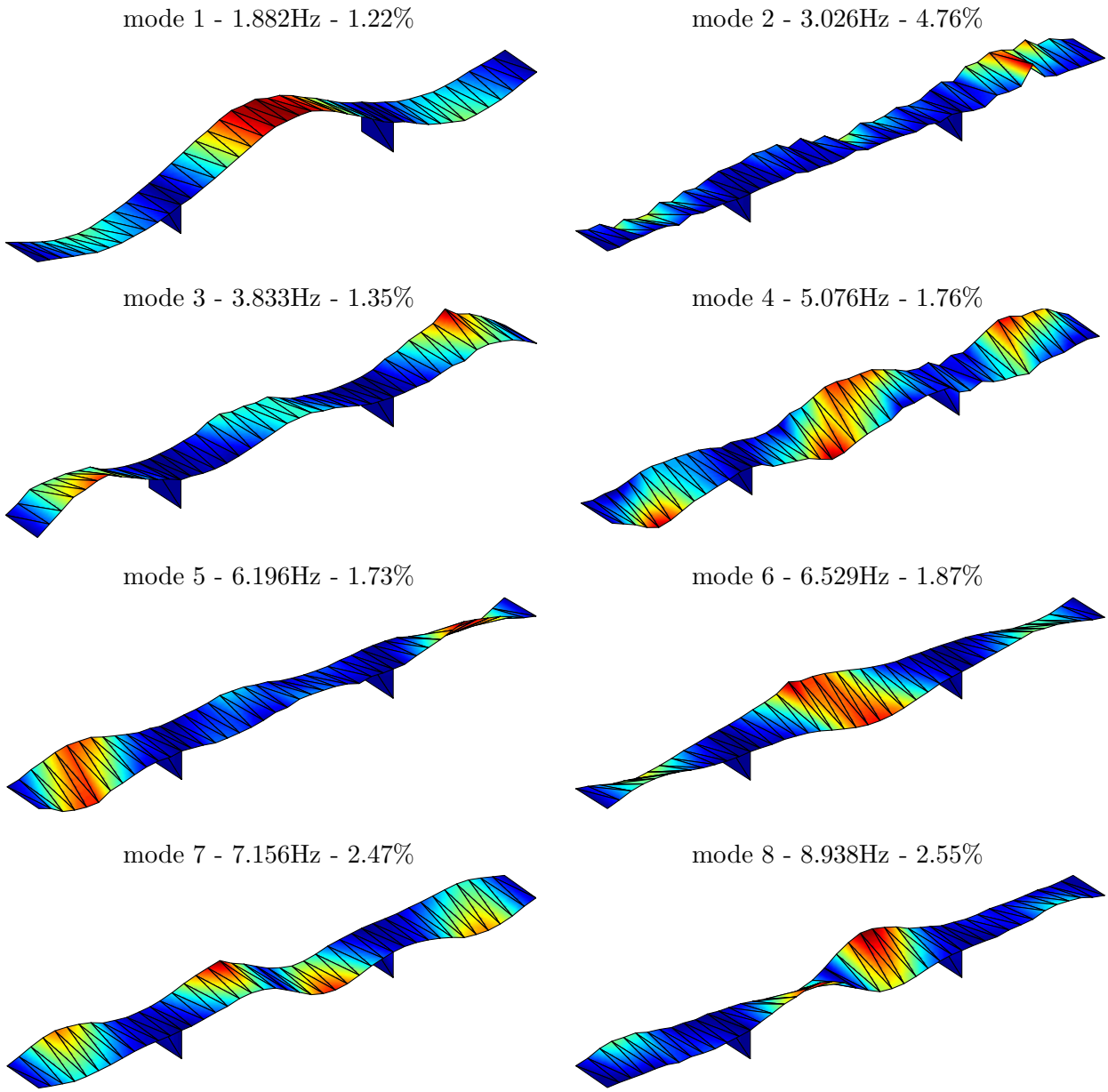


Figure 4.46: Tutorial 2: Modes obtained with CSI/ref.

4.2.7 Conclusions

In this tutorial, Experimental Modal Analysis with CMIF and deterministic pLSCF identification, and Combined Modal Analysis with CSI/ref identification, were addressed. A case study was presented, namely the modal analysis of the B15 bridge that overpasses the E19 highway between Brussels and Antwerp, Belgium. Table 4.6 compares the eigenfrequencies, damping ratios, modal phase collinearities (MPC), and mean phases (MP), obtained with all methods. Note that for the CMIF method, no modal damping ratios are available and that the modes can not be mass-normalized, so that the MP is not plotted for this method. From the table, it can be seen that the MPC values are generally higher for the CSI/ref estimates, indicating a more accurate mode shape estimate. The eigenfrequency estimates for the CMIF and CSI/ref methods are very close, but the damping ratios for the pLSCF method are

mode	CMIF		pLSCF				CSI/ref			
nr.	f [Hz]	MPC [–]	f [Hz]	ξ [%]	MPC [–]	MP [°]	f [Hz]	ξ [%]	MPC [–]	MP [°]
1	1.943	0.99	1.884	0.89	0.97	4	1.882	1.22	0.99	2
2	–	–	–	–	–	–	3.026	4.76	0.63	23
3	3.842	0.92	3.830	1.14	0.74	26	3.833	1.35	0.97	4
4	–	–	5.081	0.59	0.80	21	5.076	1.76	0.91	9
5	–	–	6.181	1.31	0.89	10	6.196	1.73	0.98	6
6	6.504	0.82	6.525	1.74	0.98	5	6.529	1.87	0.98	20
7	7.155	0.58	7.164	2.36	0.97	9	7.156	2.47	0.97	29
8	8.947	0.80	8.933	2.23	0.97	8	8.937	2.55	0.90	46
9	–	–	13.266	2.29	0.79	16	–	–	–	
10	–	–	16.713	2.29	0.68	22	–	–	–	

Table 4.6: Tutorial 2: Modes, identified with deterministic pLSCF and CSI/ref: eigenfrequencies, damping ratios, modal phase collinearities (MPC), and mean phases (MP).

systematically smaller; this is probably because the damping ratio estimates of pLSCF are biased[15]. With the CMIF method, only a limited number of modes could be estimated and the eigenfrequency estimates are rather poor, due to the rather coarse frequency resolution (0.163Hz). The MP values are very low for the first mode, indicating that the imaginary part of the mass-normalized mode shape is close to zero. For the higher modes, the MP values differ between both methods.

Using CSI/ref, it was possible to obtain more accurate information about modes that were less well excited by the drop weight forces (modes 2 and 3) than with deterministic pLSCF. This is probably due to two reasons:

1. Deterministic pLSCF is a deterministic system identification method, so the resulting modal analysis type is EMA, while CSI/ref is a combined deterministic-stochastic system identification method, so the resulting modal analysis type is OMAX. As an OMAX analysis does not consider ambient excitation as noise but as valuable excitation, it is capable to detect modes that are not (well) excited by the measured forces;

2. The CSI/ref method yields strongly consistent system estimates, while the pLSCF results in (slightly) inconsistent estimates.

On the other hand, using deterministic pLSCF, some higher modes could easily be detected due to the very clear stabilization diagram, while with CSI/ref this operation would be troublesome due to the unclear stabilization diagram. So one can conclude that deterministic pLSCF and CSI/ref complete each other well.

The CMIF method is suited for quick, rough estimates, but it should be applied with care since modes that are less well excited may be missed.

[15]The fact that pLSCF results in biased damping ratio estimates has been demonstrated through numerical simulation studies; see, e.g., [11].

4.3 Roving hammer testing of a rib-stiffened plate

In this tutorial, the capabilities of using MACEC in batch mode are explored. The objective is to identify modal characteristics from roving hammer test data.

4.3.1 Roving hammer testing

In a roving hammer test, dynamic reciprocity is employed so as to reduce the total time of testing. Using the dynamic (or Betti-Rayleigh) reciprocity theorem, it can be shown that the frequency response function $\mathbf{H}(\omega)$ between structural displacement outputs and force inputs decomposes as

$$\mathbf{H}(\omega) = \sum_j \frac{q_j \boldsymbol{\phi}_{j\mathrm{v}} \boldsymbol{\phi}_{j\mathrm{e}}^{\mathrm{T}}}{\mathrm{i}\omega - \lambda_j}, \tag{4.1}$$

where λ_j denotes the continuous-time eigenvalue of mode j, $\boldsymbol{\phi}_j$ the corresponding mode shape vector, and q_j the corresponding modal scaling factor. The parts of the mode shape vector corresponding to the output and input DOFs are denoted as $\boldsymbol{\phi}_{j\mathrm{v}}$ and $\boldsymbol{\phi}_{j\mathrm{e}}$, respectively. The decomposition also holds for the force-acceleration FRF when q_j is redefined.

Suppose now that a limited number of accelerometers and a hammer are available for modal testing, and that the mode shapes need to be identified in a large number of DOFs. The dynamic reciprocity principle can then be employed to achieve this in an elegant way as follows. First, the accelerometers are fixed to the structure at those DOFs where normally the loads would be applied. Then, a forced vibration test is performed, in which the hammer force is applied at the first mode shape DOF of interest. From the recorded force-acceleration data, a first FRF, denoted by $\mathbf{H}_1$, is estimated. Subsequently, the test is repeated, but with the hammer force applied at the second mode shape DOF of interest, resulting in a second FRF, denoted by $\mathbf{H}_2$. By further repeating the test, each time roving the hammer, additional FRFs are obtained. Note that roving the hammer from one DOF to another usually requires a minimal effort, the time in between two tests is often negligible compared to the duration of a test.

Since all FRFs that are obtained from a roving hammer test have the same output DOFs but different input DOFs, they can be combined as follows:

$$\mathbf{H}_{\mathrm{comb}}(\omega) = \begin{bmatrix} \mathbf{H}_1 & \mathbf{H}_2 & \dots & \mathbf{H}_{\mathrm{h}} \end{bmatrix} \tag{4.2}$$

$$= \sum_j \frac{q_j \boldsymbol{\phi}_{j\mathrm{v}} \begin{bmatrix} \boldsymbol{\phi}_{j\mathrm{e}1}^{\mathrm{T}} & \boldsymbol{\phi}_{j\mathrm{e}2}^{\mathrm{T}} & \boldsymbol{\phi}_{j\mathrm{eh}}^{\mathrm{T}} \end{bmatrix}}{\mathrm{i}\omega - \lambda_j} \tag{4.3}$$

$$= \sum_j \frac{q_j \boldsymbol{\phi}_{j\mathrm{v}} \boldsymbol{\phi}^{\mathrm{T}}}{\mathrm{i}\omega - \lambda_j}, \tag{4.4}$$

where $\boldsymbol{\phi}$ denotes the full mode shape vector at all DOFs of interest. Taking the transpose results in the reciprocal FRF

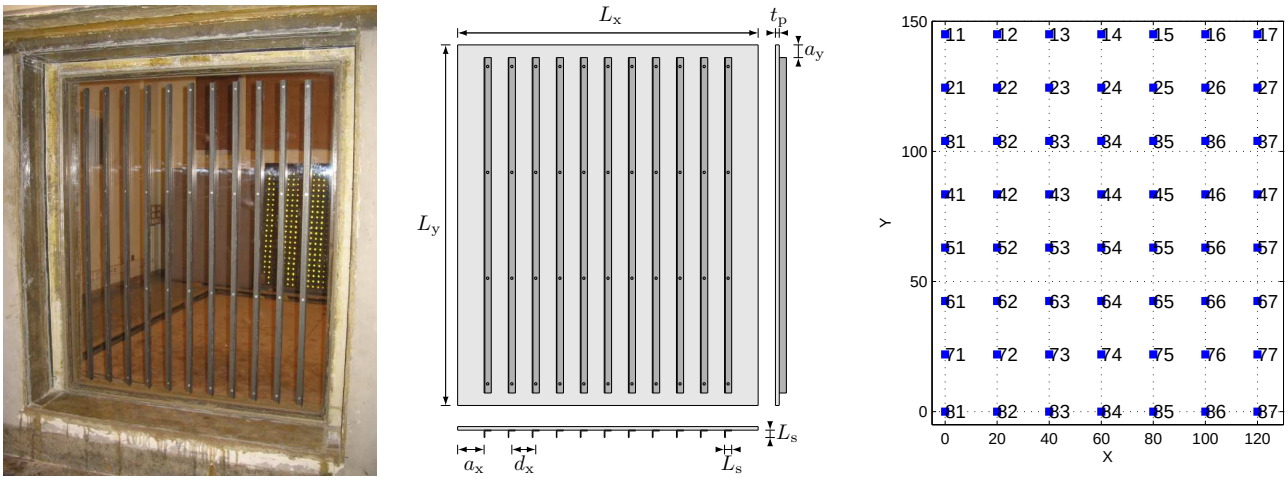
$$\mathbf{H}(\omega) = \mathbf{H}_{\mathrm{comb}}^{\mathrm{T}}(\omega) \tag{4.5}$$

$$= \sum_j \frac{q_j \boldsymbol{\phi} \boldsymbol{\phi}_{j\mathrm{v}}^{\mathrm{T}}}{\mathrm{i}\omega - \lambda_j}, \tag{4.6}$$

from which the natural frequencies, damping ratios and full mode shapes of interest can be identified using e.g. the peak picking or pLSCF methods as detailed in the previous tutorial (see section 4.2.5).

4.3.2 Rib-stiffened plate

The structure that is considered in this tutorial consists of a base plate made of polymethyl metacrylate (PMMA), to which 11 steel L30 stiffeners are attached (see Fig. 4.47). The base plate has a width of $L_x = 1.25\,\text{m}$, a height of $L_y = 1.5\,\text{m}$ and a thickness of $t_p = 15\,\text{mm}$. The steel L-shaped stiffeners have an outer leg length of $L_s = 30\,\text{mm}$ and a thickness of $t_s = 3\,\text{mm}$. The center-to-center spacing between the stiffeners is $d_x = 100\,\text{mm}$, the distance between a vertical edge of the plate and a vertical edge of the closest stiffener is $a_x = 110\,\text{mm}$, and the distance between a horizontal edge of the plate and the closest end section of a stiffener is $a_y = 52.5\,\text{mm}$. The stiffeners are both glued to the base plate over their entire length, and additionally screwed to the base plate at four points.



(a) (b) (c)

Figure 4.47: Tutorial 3: Rib-stiffened plate: (a) as mounted in a transmission opening of the acoustics laboratory of KU Leuven, (b) sketch indicating the geometrical definitions, and (c) sketch indicating the measurement DOFs (right).

An experimental modal analysis has been performed when the stiffened plate was mounted in a transmission opening between two reverberation rooms of the building acoustics laboratory of KU Leuven (see Fig. 4.47a). A roving hammer test was carried out, during which the plate was excited with an impact hammer at 56 different positions, and the response of the plate was measured with accelerometers at 5 fixed positions. The measurement locations are defined in Fig. 4.47c. The fixed accelerometer positions have numbers 31, 36, 43, 54 and 75.

The measurement data can be found in the `..\spice\tutorials\tutorial3\data` directory. There are 56 data files, one for each impact hammer test. The file number corresponds to the impact hammer location as defined in Fig. 4.47c.

Again, the first thing to do would be to build the geometry in MACEC by creating a grid file and a beam or surface file. As this step has already been treated in detail in the first tutorial (see section 4.1.2), it is not explained in detail here. The grid and surface files have been prepared on beforehand. They can be found in the `..\spice\tutorials\tutorial3\calculations` directory.

4.3.3 Processing the measured signals into an FRF matrix

1. The measurement data can be loaded into Matlab using the MACEC GUI, as detailed in the previous tutorials. However, in the present tutorial, we will achieve this without the GUI, by using MACEC in batch mode. First, we create a blank Matlab script by clicking on `New Script` in the Matlab main window. Save this script file in the `..\spice\tutorials\tutorial3\calculations` directory as `logfile_tutorial3.m`, and create a header for the logfile, e.g.

```
% Tutorial 3: roving hammer test of a rib-stiffened plate
clear all, clc   % clear the Matlab memory and command window
```

2. Then we load the measurement data of the first hammer test into Matlab. From section 5.2.3 of this manual, it is clear that this can be achieved with the INPUT2MCSIGNAL function of MACEC. Let us add the following line of code to our logfile:

```
x = input2mcsignal('../data/Ex11.DDF','ddf',1);  % load the measurement data
```

   The resulting variable X is a cell of MCSIGNAL objects. In this case, there is only a single MCSIGNAL object in the cell because only a single measurement file was converted. Therefore we can redefine X:

```
x = x{1};  % redefine x as an mcsignal object instead of a cell
```

3. The measurement data have now been converted into an MCSIGNAL object, yet they still need to be converted from electrical potential to force and acceleration units. Furthermore, the type of data (force or acceleration) needs to be defined for each measurement channel, and labels need to be assigned to the different measurement channels. As this conversion will be performed for an MCSIGNAL object, we need one of the MACEC functions that are listed in section 5.1 to achieve this. After inspecting the different functions available, it is clear that the VOLT2XA function is needed here. A correct use of the function results in the following Matlab code (see section 5.1.50 for more details on the VOLT2XA function):

```
sens = [0.0019806;0.010503;0.010003;0.0096465;0.010697;0.0096975];
                                % sensor sensitivities
ampl = ones(6,1);               % amplification factors ...
db   = zeros(6,1);              %         ... are not in dB
types = cell_(strvcat('force','acc','acc','acc','acc','acc')); % data types
labels = cell_(strvcat('hammer','s1','s2','s3','s4','s5'));    % channel labels
predat = volt2xa(x,sens','ampl','db','types','labels');            % signal conversion
```

4. The next stage is preprocessing these converted data. The preprocessing steps all involve operations on the PREDAT variable, which is an MCSIGNAL object. Therefore, all preprocessing operations will involve MACEC functions that are listed in section 5.1. For example, by adding the following command to the logfile and running the logfile again, the static (or DC) component is removed from the data:

```
predat = detrend(predat,'constant');    % remove DC component
```

5. The data have been sampled at 4096 Hz, as can be verified by executing the following command in the Matlab main window:

```
predat.F
```

However, the frequency range of interest in this analysis is $0 - 300$ Hz. In order to concentrate the analysis on the frequency range of interest, we will re-sample the data at a rate which is five times lower, resulting in a new sampling frequency of $4096/5 = 819.2$ Hz and a new Nyquist frequency of 409.6 Hz. This can be achieved with the DECIMATE function (see section 5.1.6). This function employs an eighth-order low-pass Chebychev Type I filter with a cutoff frequency of 0.8 times the new Nyquist frequency[16] in both the forward and reverse directions to remove all phase distortion, before re-sampling the data. The function is applied by adding the following command to the logfile:

```
predat = decimate(predat,5);            % resample
```

6. At any stage of data processing, it is a good practice to look at the data in both the time and frequency domain in order to detect any anomalies or to verify the effect of a signal processing step. In this case the data have been checked at the time of acquisition, yet it is a good idea to perform a final check before going to the system identification stage. MACEC has powerful functions for plotting data that are related to an MCSIGNAL object. For example, plotting the time history and the amplitude of the related Fourier transform of the third channel can be easily achieved with the following commands, which make use of the AXESPLOT function of MACEC (see section 5.1.3):

```
figure, axesplot('t',3,predat)
figure, axesplot('f',3,predat,'logy',1)
```

The results are shown in Fig. 4.48a and Fig. 4.48b. Note that the correct units and axis labels are immediately applied. It can be noted that a low-frequency drift distorts the signal. Therefore, the following command lines are added to the logfile, resulting in a high-pass filtering of the data with a cut-on frequency of 10 Hz ($= 409.6$ Hz $\times$ 0.024414):

```
[b,a] = butter([4],0.024414,'high');    % define high-pass filter
predat = filtfilt(1:6,predat,b,a);      % apply the filter
```

The time history and Fourier amplitude spectrum of the third channel, after high-pass filtering, can be plotted again in the same way as before; the results are shown in Fig. 4.48c and Fig. 4.48d, respectively. The influence of the high-pass filter is clearly visible.

7. After completing the signal preprocessing stage, an FRF can be estimated from the processed data using the nonparametric H_1 estimator[17]. This estimator has been implemented into the

[16] In the present application this amounts to a cutoff frequency of 327.68 Hz

[17] See [11, Sec. 5.2.2] for a detailed description and an extensive discussion of the H_1 estimator. In MACEC, the periodogram variant has been implemented as it enables to estimate the variance errors of the FRF estimates.

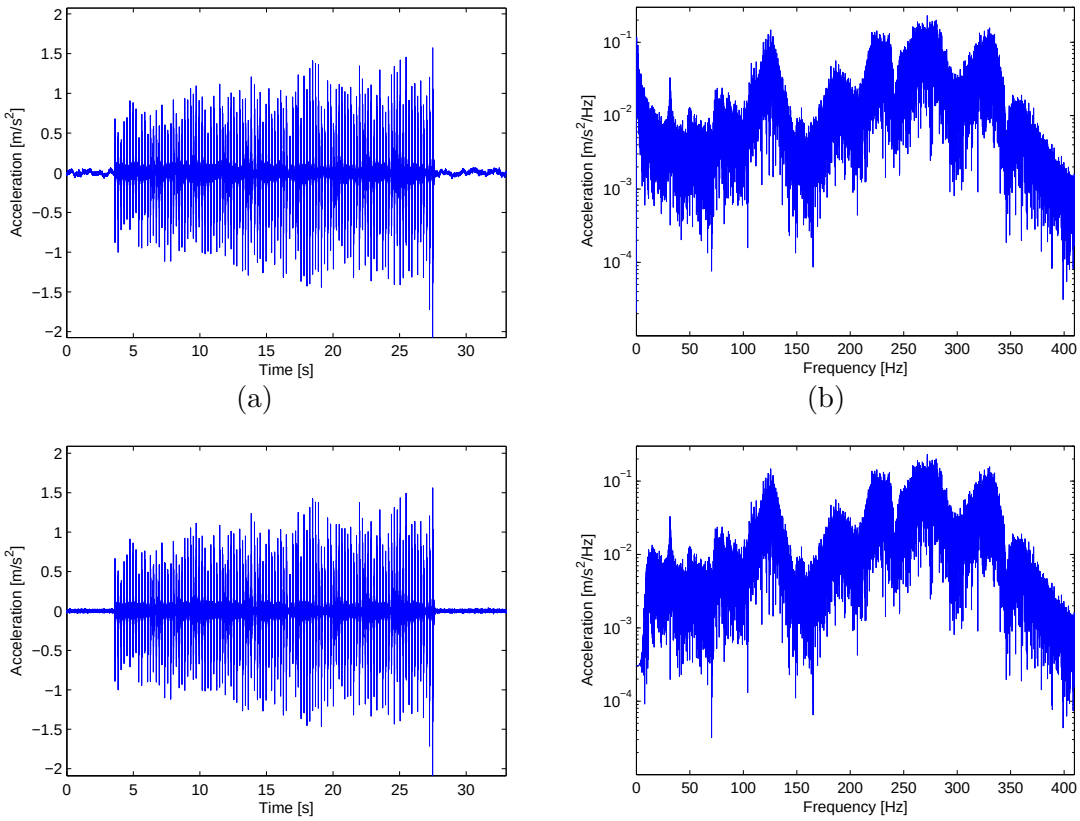


Figure 4.48: Tutorial 3: Rib-stiffened plate: (a,c) time history and (b,d) Fourier amplitude spectrum of the third channel of the first measurement setup, (a,b) after removing the DC component and re-sampling, and (c,d) after additional high-pass filtering with a cut-on frequency of 10 Hz.

MACEC function H1ESTIMATE (see section 5.5.1). Because the H_1 estimator relies on an averaging process, the input and output time data are divided into a number of blocks of equal length. When choosing the number of blocks, a trade-off needs to be made: a small number of blocks may result in significant variance errors on the FRF estimate due to insufficient averaging, while a large number of blocks may result in significant bias errors due to leakage [11, Sec. 5.2.2]. A crude rule of thumb would be to choose the number of data blocks such that the number of samples in one data block is not smaller than $2000 - 5000$ samples. The total number of data samples available in the `predat` variable can be obtained by executing the following command in the Matlab main window:

```
predat.N
```

A total of 27034 samples are available. When dividing the data into 8 blocks, there are 3379 samples per block. The following command line can then be added to the logfile in order to estimate the FRF:

```
[H1,f] = H1estimate(predat,8,'rect',1,2:6,'fs');  % estimate FRF
```

The third argument indicates that a rectangular window is applied (which amounts to no windowing in this case). The fourth argument indicates that the first channel contains the force, and the fifth argument indicates that channels 2 to 6 contain the response. The last argument

indicates that the FRF needs to be estimated for all frequencies up to the sampling frequency. This is necessary because in a later stage, the pLSCF method will be employed for estimating a parametric input-output model (i.e., a right matrix fraction description model). When peak picking is applied, the last argument can be changed into `'fn'`; in that case FRF is estimated only for frequencies up to the Nyquist frequency.

8. The previous procedure can now be employed for estimating the FRF corresponding to each hammer position, and for gathering the corresponding estimates into a big overall FRF matrix estimate. At this stage, the true power of executing MACEC functions in batch mode becomes apparent, as only minor modifications to the Matlab script that has been built up so far are necessary for processing all hammer test data files, instead of just the single file that has been processed so far. Basically, a for loop needs to be written around the current script together with a few related changes, resulting in the Matlab script of Fig. 4.49. The only additional difficulty is caused by the fact that not all test data sequences have the same length, so an additional signal processing step is necessary to ensure that all partial FRF estimates have the same number of frequency lines. This has been achieved by adding the following command line:

```
predat = trim(predat,1,116736);          % apply time window
```

The effect is that, for each test, a time window is applied such that only the first 116736 samples are retained. As a result, there are 2918 frequency lines in the overall FRF matrix estimate.

4.3.4 System identification using the pLSCF method

After estimating the overall FRF matrix using the nonparametric H_1 estimator, the modal characteristics of interest could be obtained immediately using the peak picking or CMIF approach. However, parametric system identification generally yields much more accurate results [11], and therefore an additional parametric system identification step is performed here. The pLSCF method is employed for fitting a parametric, right matrix fraction description (or RMFD) model to the nonparametric FRF data. System identification with the pLSCF method using the MACEC GUI has already been treated in the two previous tutorials. In this tutorial, the batch mode of MACEC is explored.

9. An RMFD description can be estimated from nonparametric FRF data using the MACEC function RMFDCALC (see section 5.5.4). One of the parameters that need to be specified by the user is the model order range. Please note that the model order of an RMFD model is not equal to that of a state-space model: the model order of a state space model equals the model order of the corresponding RMFD model multiplied with the number of inputs (amounting to 5 in this case) [11, Sec. 2.2.7]. Since the true system order is unknown, a wide range of RMFD model orders is chosen here: from 2 to 100 in steps of 2. The resulting lines of code read

```
% estimate RMFD description using the pLSCF method
rmfd = RMFDcalc(Htot,'pLSCF',[2:2:100],1:nDOFs,nDOFs+[1:5],[],f,f(1),f(end/2));
```

Note that the RMFDCALC function also asks for output and input channel numbers, these are the fourth and fifth arguments of the function, respectively. Because the FRF was estimated from different data files, dummy numbers are assigned here. The outputs are given channel numbers 1 to 56, and the inputs numbers 57 to 61.

```
% Tutorial 3: roving hammer test of a rib-stiffened plate
clear all, close all, clc

DOFs = repmat(1:7,[1 8])+kron(1:8,10*ones(1,7)); % list of DOF numbers
nDOFs = length(DOFs);   % total number of DOFs
Htot = zeros(nDOFs,5,2918); % overall FRF matrix
for par = 1:nDOFs % loop over all hammer tests
    hDOF = DOFs(par);   % DOF at which the hammer is applied
    file = ['../data/Ex' num2str(hDOF) '.DDF'];  % current data file
    x = input2mcsignal(file,'ddf');   % load the measurement data
    x = x{1};  % redefine x as an mcsignal object instead of a cell
    sens = [0.0019806;0.010503;0.010003;0.0096465;0.010697;0.0096975];
                                      % sensor sensitivities
    ampl = ones(6,1);                 % amplification factors ...
    db   = zeros(6,1);                %         ... are not in dB
    types = cell_(strvcat('force','acc','acc','acc','acc','acc')); % data types
    labels = cell_(strvcat('hammer','s1','s2','s3','s4','s5'));    % channel labels
    predat = volt2xa(x,sens',ampl',db',types',labels');       % signal conversion
    predat = trim(predat,1,116736);         % apply time window
    predat = detrend(predat,'constant');    % remove DC component
    predat = decimate(predat,5);            % resample
    [b,a] = butter([4],0.024414,'high');    % define high-pass filter
    predat = filtfilt(1:6,predat,b,a);      % apply the filter
    [H1,f] = H1estimate(predat,8,'rect',1,2:6,'fs'); % estimate FRF
    Htot(par,:,:) = H1;    % apply reciprocity: switch inputs and outputs
end
```

Figure 4.49: Tutorial 3: Matlab script, making use of MACEC functions in batch, for processing 56 roving hammer test files into a big overall FRF matrix estimate.

10. Before advancing to the modal analysis stage, node numbers, measurement directions and measurement quantities need to be attached to the different channels. This can be achieved with the MACEC function IDENTSEL (see section 5.5.6):

```
node_num = [DOFs, 31 43 54 36 75]';   % assign node numbers to channels
meas_dir = [repmat([0 90],[maxpar,1]); repmat([0 -90],[5,1])]; % measurement directions
quants = cell(maxpar+5,1);            % measured quantities
    for par = 1:maxpar, quants{par} = 'acc'; end
    for par = maxpar+1:maxpar+5, quants{par} = 'force'; end
[node_num,meas_dir,quants,chan_outin,ampl_outin,refs] = identsel(node_num,...
          meas_dir,quants,rmfd.chanselout,rmfd.chanselin,rmfd.refs);
```

11. After parametric system identification, the modal characteristics are selected from a stabilization diagram and this requires significant user interaction. Therefore, we will switch from batch mode to the MACEC GUI at this stage. In order to load the identification results into the MACEC GUI, they need to be saved first. This is achieved by adding the following line of code to our logfile:

```
save('tut3_pLSCF.mat','ampl_outin','chan_outin','meas_dir','node_num',...
          'predat','quants','refs','rmfd');
```

4.3.5 Modal analysis after deterministic pLSCF

12. We can now select the modal characteristics in a stabilization diagram. In order to do so, open the main window of the MACEC GUI, load the `tut3_pLSCF.mat` file, and press the Modal analysis button in the Modal Analysis field. The stabilization diagram that is now constructed is very clear, so the stable modes can be easily selected.

13. After selecting the stable modes from the stabilization diagram, the results can be saved and visualized as detailed in the previous tutorials.

4.3.6 Interpretation and conclusions

The set modes that you have identified will most probably include the ones that are presented in Fig. 4.50. From an inspection of the mode shapes, it is clear that they can be characterized by a two-digit combination $(n_{\mathrm{x}}, n_{\mathrm{y}})$, where n_{x} denotes the number of half wavelengths in the x-direction and n_{y} denotes the number of half wavelengths in the y-direction. Besides the modes that are presented in the figure, a few other stable modes may have been selected. However, their mode shapes cannot be given a clear physical interpretation and these modes do not show up when modal characteristics are numerically computed from a detailed finite element model of the rib-stiffened plate. This indicates that they additional stable modes are an artefact of the modal analysis process. One may therefore conclude that modal testing is often not straightforward even in laboratory conditions, and that the results should be carefully analyzed and interpreted.

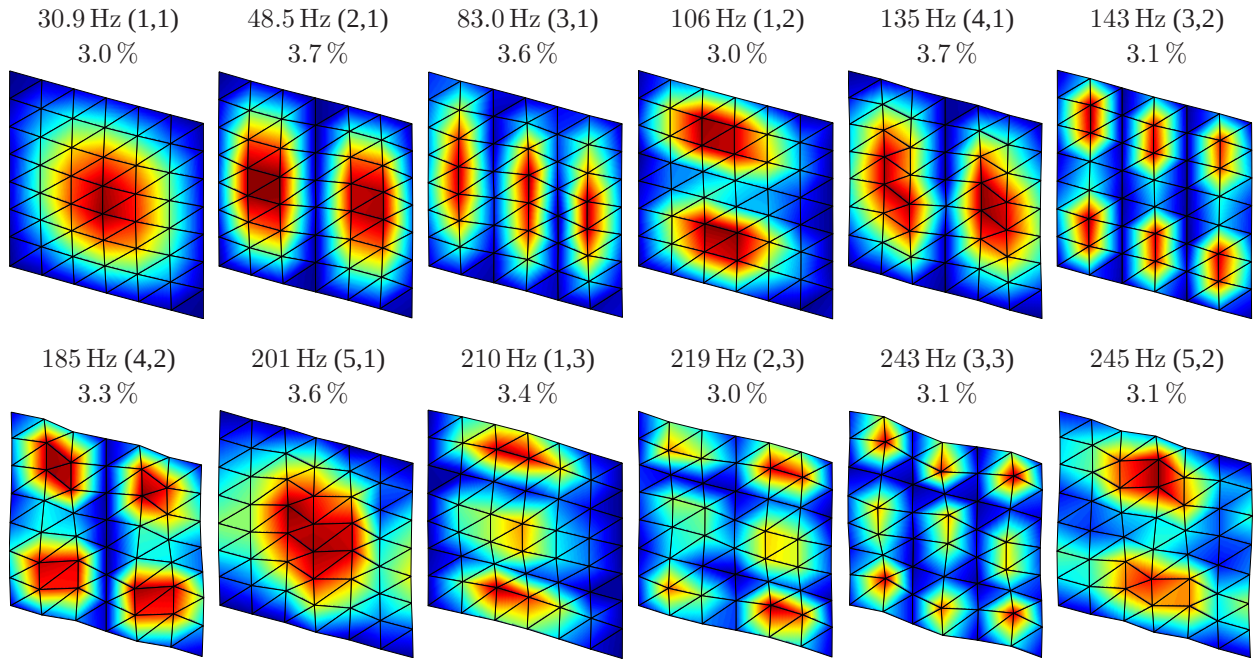


Figure 4.50: Tutorial 3: rib-stiffened plate: eigenfrequencies, damping ratios and mode shapes obtained with deterministic pLSCF.

Chapter 5

Overview of MACEC functions

In this chapter, all individual functions of the MACEC program are described. This enables you to

- use the logfile and to detect possible errors in it;
- run separate MACEC functions in batch.

The functions are grouped in sections according to the different subdirectories in the `spice32` directory. Within each subdirectory, the functions are described in alphabetical order.

5.1 @mcsignal

5.1.1 axesACPSDplot

```
AXESACPSDPLOT   Plot multi channel signal in the current axes.
   AXESACPSDPLOT(domain,ich1,x,filename,'KeyName','KeyValue',...) plots
   the autocorrelation [domain='ac'] or the power spectral density
   [domain='psd'] using Welch's method of the selected channels ich1 of
   the mcsignal x.  Key options can be specified to fine tune the plot.
   These options are interpreted by the present function, and they are
   passed to the global function NEWFIG which is used to open a new figure
   window.  The values of the key options interpreted by the present
   function may be scalars / single strings, or vectors / cell arrays of
   strings.  In the first case, the same value is employed for all
   channels, while in the second case, values for all of the mcsignal's
   channels (not only the selected channels ich1) have to be provided.

   domain       'ac' to plot autocorrelation, 'f' to plot power spectral
                density.
   ich1         Channels to plot.  Default: all channels.
   x            mcsignal object.
   filename     File to save the plot to, using the global function SAVEFIG.
                Can be omitted if you do not want to save the plot.
   'Windowtype'   In case of a PSD plot, the window type has to be
                specified. Valid choises are 'rectwin' and 'hanning'.
                Default: rectangular window ('rectwin').
```

```
'Windowlength' In case of a PSD plot, the window length (in data points)
            has to be specified. Default: n_(x)
'Overlap'   In case of a PSD plot, the overlap length (in data points)
            has to be specified. Default: 0
'AddToPlot' Command evaluated after all built in plotting routines, as to
            add elements to the figure, e.g. equal human response curves.
'AxesSize'  Size of the axes in centimeters.  See NEWFIG for more info.
'Blank'     Do not plot the curve, plot only the axes and the labels.
'Data'      Value is assigned to the local variable 'Data', which can be
            used by the 'AddToPlot' command.  If a cell array is provided,
            the contents of cell i is assigned to the local variable 'Data'
            when plotting channel i.
'ix'        First index of the time interval to plot.
            Default: 1
'jx'        Last index of the time interval to plot.
            Default: x.N
'logx'      Logarithmic scale for first axis or not
            Default: 0
'logy'      Logarithmic scale for first axis or not
            Default: 0
'Language'  Label language.
            'en' or 'nl'
            Default: 'en'
'Legend'    Default: no legend.  See also LEGEND.
'LegendLoc' Legend location.  Default: 'Best'.  See also LEGEND.
'LineWidth' Default: 0.5 for domain 't' and 'f', and 1.5 for domain 'b'.
'Margins'   Margins between the axes and the bounding box, in centimeters.
            See NEWFIG for more info.
'NXTick'    Number of XTicks (approximately, see the global function TICK).
            Default: auto.
'NYTick'    Number of YTicks (approximately, see the global function TICK).
            Default: auto.
'XLabel'    X-axis label.
            Default: defined using MCSIGNAL/LABEL
'YLabel'    Y-axis label.
            Default: defined using MCSIGNAL/LABEL
'XMin'      X-axis scaling: minimum value.
            Default: left side of the plotted curve.
'XMax'      X-axis scaling: maximum value.
            Default: right side of the plotted curve.
'YMin'      Y-axis scaling: minimum value.
            Default: defined as a function of the plotted curve Y(X):
            -max(abs(Y)) for domain 't', 0 for domain 'f' and min(Y) for
            domain 'b'.
'YMax'      Y-axis scaling: maximum value.
            Default: defined as a function of the plotted curve Y(X):
            max(abs(Y)) for domain 't' and max(Y) for domains 'f' and 'b'.
'Color'     Color of the plot. Default: 'b' (blue). Other valid
            parameters: 'g' (green), 'r' (red), 'c' (cyan), 'm'
            (magenta), 'y' (yellow), 'k' (black) and 'w' (white)
```

```
   See also NEWFIG, SAVEFIG, MCSIGNAL/TPLOT, MCSIGNAL/FPLOT, MCSIGNAL/BPLOT.
```

5.1.2 axesCPSDFRFCOHplot

```
AXESCPSDFRFCOHPLOT   Plot multi channel signal in the current axes.
   AXESCPSDFRFCOHPLOT(domain,refch,ich1,x,filename,'KeyName','KeyValue',...) plots
   the cross power spectral density [domain='cpsd'], the frequency response
   function [domain='frf'] or the coherence function [domain='coh'] of
   the mcsignal x.  Key options can be specified to fine tune the plot.
   These options are interpreted by the present function, and they are
   passed to the global function NEWFIG which is used to open a new figure
   window.  The values of the key options interpreted by the present
   function may be scalars / single strings, or vectors / cell arrays of
   strings.  In the first case, the same value is employed for all
   channels, while in the second case, values for all of the mcsignal's
   channels (not only the selected channels ich1) have to be provided.

   domain       'cpsd' to plot cross power spectral density, 'frf' to plot
                frequency response function and 'coh' to plot coherence
                function.
   refch        Reference channel.
   ich1         Channels to plot.  Default: all channels.
   x            mcsignal object.
   filename     File to save the plot to, using the global function SAVEFIG.
                Can be omitted if you do not want to save the plot.
   'Windowtype' In case of a PSD plot, the window type has to be
                specified. Valid choises are 'rectwin' and 'hanning'.
                Default: rectangular window ('rectwin').
   'Windowlength' In case of a PSD plot, the window length (in data points)
                has to be specified. Default: n_(x)
   'Overlap'    In case of a PSD plot, the overlap length (in data points)
                has to be specified. Default: 0
   'AddToPlot'  Command evaluated after all built in plotting routines, as to
                add elements to the figure, e.g. equal human response curves.
   'AxesSize'   Size of the axes in centimeters.  See NEWFIG for more info.
   'Blank'      Do not plot the curve, plot only the axes and the labels.
   'Data'       Value is assigned to the local variable 'Data', which can be
                used by the 'AddToPlot' command.  If a cell array is provided,
                the contents of cell i is assigned to the local variable 'Data'
                when plotting channel i.
   'ix'         First index of the time interval to plot.
                Default: 1
   'jx'         Last index of the time interval to plot.
                Default: x.N
   'logx'       Logarithmic scale for first axis or not
                Default: 0
   'logy'       Logarithmic scale for first axis or not
                Default: 0
```

```
   'Language'   Label language.
                'en' or 'nl'
                Default: 'en'
   'Legend'     Default: no legend.  See also LEGEND.
   'LegendLoc'  Legend location.  Default: 'Best'.  See also LEGEND.
   'LineWidth'  Default: 0.5 for domain 't' and 'f', and 1.5 for domain 'b'.
   'Margins'    Margins between the axes and the bounding box, in centimeters.
                See NEWFIG for more info.
   'NXTick'     Number of XTicks (approximately, see the global function TICK).
                Default: auto.
   'NYTick'     Number of YTicks (approximately, see the global function TICK).
                Default: auto.
   'XLabel'     X-axis label.
                Default: defined using MCSIGNAL/LABEL
   'YLabel'     Y-axis label.
                Default: defined using MCSIGNAL/LABEL
   'XMin'       X-axis scaling: minimum value.
                Default: left side of the plotted curve.
   'XMax'       X-axis scaling: maximum value.
                Default: right side of the plotted curve.
   'YMin'       Y-axis scaling: minimum value.
                Default: defined as a function of the plotted curve Y(X):
                -max(abs(Y)) for domain 't', 0 for domain 'f' and min(Y) for
                domain 'b'.
   'YMax'       Y-axis scaling: maximum value.
                Default: defined as a function of the plotted curve Y(X):
                max(abs(Y)) for domain 't' and max(Y) for domains 'f' and 'b'.

   See also NEWFIG, SAVEFIG, MCSIGNAL/TPLOT, MCSIGNAL/FPLOT, MCSIGNAL/BPLOT.
```

5.1.3 axesplot

```
AXESPLOT   Plot multi channel signal in the current axes.
   AXESPLOT(domain,ich1,x,filename,'KeyName','KeyValue',...) plots the time
   history [domain='t'], the frequency content (continuous Fourier transform)
   [domain='f'] or the one third octave band spectrum of the RMS value
   [domain='b'] of the selected channels ich1 of the mcsignal x.  Key options
   can be specified to fine tune the plot.  These options are interpreted by
   the present function, and they are passed to the global function NEWFIG
   which is used to open a new figure window.  The values of the key options
   interpreted by the present function may be scalars / single strings, or
   vectors / cell arrays of strings.  In the first case, the same value is
   employed for all channels, while in the second case, values for all of the
   mcsignal's channels (not only the selected channels ich1) have to be
   provided.

   domain       't' to plot time history, 'f' to plot frequency content, or 'b'
                to plot the one thirds octave band spectrum of the RMS value.
   ich1         Channels to plot.  Default: all channels.
```

x	mcsignal object.
filename	File to save the plot to, using the global function SAVEFIG. Can be omitted if you do not want to save the plot.
'AddToPlot'	Command evaluated after all built in plotting routines, as to add elements to the figure, e.g. equal human response curves.
'AxesSize'	Size of the axes in centimeters. See NEWFIG for more info.
'Blank'	Do not plot the curve, plot only the axes and the labels.
'Data'	Value is assigned to the local variable 'Data', which can be used by the 'AddToPlot' command. If a cell array is provided, the contents of cell i is assigned to the local variable 'Data' when plotting channel i.
'ix'	First index of the time interval to plot. Default: 1
'jx'	Last index of the time interval to plot. Default: x.N
'logx'	Logarithmic scale for first axis or not Default: 0
'logy'	Logarithmic scale for first axis or not Default: 0
'Language'	Label language. 'en' or 'nl' Default: 'en'
'Legend'	Default: no legend. See also LEGEND.
'LegendLoc'	Legend location. Default: 'Best'. See also LEGEND.
'LineWidth'	Default: 0.5 for domain 't' and 'f', and 1.5 for domain 'b'.
'Margins'	Margins between the axes and the bounding box, in centimeters. See NEWFIG for more info.
'NXTick'	Number of XTicks (approximately, see the global function TICK). Default: auto.
'NYTick'	Number of YTicks (approximately, see the global function TICK). Default: auto.
'XLabel'	X-axis label. Default: defined using MCSIGNAL/LABEL
'YLabel'	Y-axis label. Default: defined using MCSIGNAL/LABEL
'XMin'	X-axis scaling: minimum value. Default: left side of the plotted curve.
'XMax'	X-axis scaling: maximum value. Default: right side of the plotted curve.
'YMin'	Y-axis scaling: minimum value. Default: defined as a function of the plotted curve Y(X): -max(abs(Y)) for domain 't', 0 for domain 'f' and min(Y) for domain 'b'.
'YMax'	Y-axis scaling: maximum value. Default: defined as a function of the plotted curve Y(X): max(abs(Y)) for domain 't' and max(Y) for domains 'f' and 'b'.

See also NEWFIG, SAVEFIG, MCSIGNAL/TPLOT, MCSIGNAL/FPLOT, MCSIGNAL/BPLOT.

5.1.4 bplot

```
BPLOT   Plot multi channel signal 1/3 octave band RMS spectrum.
   BPLOT(ich1,x,filename,'KeyName','KeyValue',...) executes the function
   PLOT('b',ich1,x,filename,'KeyName','KeyValue',...).

   Use x.bplot{ich1}(filename,'KeyName','KeyValue',...) to access this function.

   See also MCSIGNAL/PLOT.
```

5.1.5 concatenate

```
MCSIGNAL   Multi channel time signal (unity time = 1 s) constructor.
   y = MCSIGNAL(s1,s2,...) concatenates the time histories of the mcsignals
   s1, s2,... to a single mcsignal. The number of channels must be the
   same for all mcsignal objects. The sampling frequency, quantity,
   sifactor and labels of y are equal to those of s1.

   s1,s2,...   mcsignal objects to assemble into 1 mcsignal.
   y           resulting mcsignal object

   See the example master files for info on the usage of the MCSIGNAL class.
```

5.1.6 decimate

```
DECIMATE   Decimate a mcsignal.
   y = DECIMATE(x,...) applies the global function
   DECIMATE(x.tdata,...) to the channels of the mcsignal x.

   x          mcsignal object.
   y          mcsignal object.

   Use y = x.decimate(...) to access this function.

   See also the global function DECIMATE.
```

5.1.7 delete

```
DELETE   Delete channels from a mcsignal.
   y = DELETE(ich1,x) deletes the channels ich1 from the mcsignal x.

   x          mcsignal object.
   ich1       Channels to delete.  Default: all channels.
   y          mcsignal object.

   Use y = x.delete{ich1} to access this function.

   See also MCSIGNAL/SELECT.
```

5.1.8 detrend

```
DETREND   Detrend a mcsignal.
   y = DETREND(ich1,x,...) applies the global function
   DETREND(x.tdata(:,ich1),...) to the specified channels ich1 of the mcsignal x.

   x          mcsignal object.
   ich1       Channels to detrend.  Default: all channels.
   y          mcsignal object.

   Use y = x.detrend{ich1}(...) to access this function.

   See also the global function DETREND.
```

5.1.9 df

```
DF   Multi channel signal frequency resolution.
   y = DF(x) returns the frequency resolution of the mcsignal x.

   x          mcsignal object.
   y          Frequency resolution [Hz].

   Use y = x.df to access this function.
```

5.1.10 dinkb

```
DINKB   Frequency weighting of a velocity mcsignal according to DIN 4150-2.
   y = DINKB(ich1,x) multiplies the frequency content of the specified
   channels ich1 of the mcsignal x with the weighting function obtained by the
   global function DINWEIGHT.  The sifactor of the specified channels is set to
   1/1000, as to obtain a signal in [mm/s].

   x          mcsignal object.
   ich1       Channels to apply the weighting to.  Default: all channels.
   y          mcsignal object.

   Use y = x.dinkb{ich1} to access this function.

   See also the global function DINWEIGHT.
```

5.1.11 dinkbf

```
DINKBF   Running effective value of a (weighted) velocity mcsignal
   according to DIN 4150-2.
   y = DINKBF(ich1,x) applies frequency weighting to the selected
   channels ich1 of the mcsignal x if necessary, so delivering a KB mcsignal.
   From this signal, the running effective value in [mm/s] is calculated
   using the global function DINKBF(x.tdata(:,ich1),x.F).
```

```
   x          mcsignal object.
   ich1       Channels to apply the operation to.  Default: all channels.
   y          mcsignal object.

   Use y = x.dinkbf{ich1} to access this function.

   See also the global function DINKBF.
```

5.1.12 dinkbtau

```
DINKBTAU   Running effective value of a (weighted) velocity mcsignal
   according to DIN 4150-2.
   y = DINKBTAU(ich1,x,...) applies frequency weighting to the selected
   channels ich1 of the mcsignal x if necessary, so delivering a KB mcsignal.
   From this signal, the running effective value in [mm/s] is calculated
   using the global function DINKBTAU(x.tdata(:,ich1),x.F,tau).

   x          mcsignal object.
   ich1       Channels to apply the operation to.  Default: all channels.
   y          mcsignal object.

   Use y = x.dinkbtau{ich1}(...) to access this function.

   See also the global function DINKBTAU.
```

5.1.13 disp

```
DISP   Display multi channel signal metadata.
   DISP(x) prints the metadata of the mcsignal x to the console window.

   x          mcsignal object.
```

5.1.14 display

```
DISPLAY   Display multi channel signal metadata.
   DISPLAY(x) prints the metadata of the mcsignal x to the console window.
   MATLAB calls this method whenever a mcsignal object is the result of a
   statement that is not terminated by a semicolon.

   x          mcsignal object.
```

5.1.15 domf

```
DOMF   Multi channel signal dominant frequency.
   y = DOMF(x) returns the dominant frequency of the mcsignal x.

   x          mcsignal object.
```

```
   y          Sampling frequency [Hz].

   Use y = x.domf to access this function.
```

5.1.16 dt

```
DT   Multi channel signal time step.
   y = DT(x) returns the time step of the mcsignal x.

   x          mcsignal object.
   y          Time step [s].

   Use y = x.dt to access this function.
```

5.1.17 f

```
F   Multi channel signal frequency vector.
   y = F(x) returns a vector containing the first positive frequencies
   corresponding to the FFT of the data stored in the mcsignal x.

   x          mcsignal object.
   y          Frequencies [Hz].

   Use y = x.f to access this function.

   See also MCSIGNAL/F2, MCSIGNAL/FFTFREQ.
```

5.1.18 f2

```
F2   Multi channel signal frequency vector.
   y = F2(x) returns a vector containing the frequencies corresponding to the
   FFT of the data stored in the mcsignal x, up to the Nyquist frequency.

   x          mcsignal object.
   y          Frequencies [Hz].

   Use y = x.f2 to access this function.

   See also MCSIGNAL/F, MCSIGNAL/FDATA2.
```

5.1.19 f_

```
F_   Multi channel signal sampling frequency.
   y = F_(x) returns the sampling frequency (*NOT* the Nyquist frequency) of
   the mcsignal x.

   x          mcsignal object.
   y          Sampling frequency [Hz].
```

```
   Use y = x.F to access this function.
```

5.1.20 f_n_

```
F_N_   Multi channel signal Nyquist frequency.
   y = F_N_(x) returns the Nyquist frequency of the mcsignal x.

   x          mcsignal object.
   y          Nyquist frequency [Hz].

   Use y = x.FN to access this function.
```

5.1.21 fdata

```
FDATA   Multi channel signal frequency content.
   y = FDATA(x) returns the frequency content of the mcsignal x, obtained by
   calling Matlab FFT which is a DFT.  An approximation of the continuous
   Fourier transform can be obtained by dividing y by the sampling frequency.

   x          mcsignal object.
   y          Frequency content.

   Use y = x.fdata to access this function.

   See also MCSIGNAL/FDATA2.
```

5.1.22 fdata2

```
FDATA   Multi channel signal frequency content.
   y = FDATA2(x) returns the frequency content of the mcsignal x up to the
   Nyquist frequency, obtained by calling Matlab FFT which is a DFT.  An
   approximation of the continuous Fourier transform can be obtained by
   dividing y by the sampling frequency.

   x          mcsignal object.
   y          Frequency content.

   Use y = x.fdata2 to access this function.

   See also MCSIGNAL/FDATA, MCSIGNAL/F2.
```

5.1.23 fftfreq

```
FFTFREQ   Multi channel signal frequency vector.
   y = FFTFREQ(x) returns a vector containing the frequencies corresponding to
   the FFT of the data stored in the mcsignal x.
   Conceptually, y = [0:df:F/2, -F/2:df:0].
```

```
   x          mcsignal object.
   y          Frequencies [Hz].

   Use y = x.fftfreq to access this function.

   See also MCSIGNAL/F and the global function FFTFREQ.
```

5.1.24 filter

```
FILTER   Filter a mcsignal.
   y = FILTER(ich1,x,B,A,...) applies the global function
   FILTER(B,A,x.tdata(:,ich1),...) to the specified channels ich1 of the
   mcsignal x.

   x          mcsignal object.
   ich1       Channels to filter.  Default: all channels.
   y          mcsignal object.

   Use y = x.filter{ich1}(B,A,...) to access this function.

   See also MCSIGNAL/FILTFILT and the global function FILTER.
```

5.1.25 filtfilt

```
FILTFILT   Filter a mcsignal.
   y = FILTFILT(ich1,x,B,A,...) applies the global function
   FILTFILT(B,A,x.tdata(:,ich1),...) to the specified channels ich1 of the
   mcsignal x.

   x          mcsignal object.
   ich1       Channels to filter.  Default: all channels.
   y          mcsignal object.

   Use y = x.filtfilt{ich1}(B,A,...) to access this function.

   See also MCSIGNAL/FILTER and the global function FILTFILT.
```

5.1.26 fplot

```
FPLOT   Plot multi channel signal frequency content.
   FPLOT(ich1,x,filename,'KeyName','KeyValue',...) executes the function
   PLOT('f',ich1,x,filename,'KeyName','KeyValue',...).

   Use x.fplot{ich1}(filename,'KeyName','KeyValue',...) to access this function.

   See also MCSIGNAL/PLOT.
```

5.1.27 integrate

```
INTEGRATE   Integrate a mcsignal.
   y = INTEGRATE(ich1,x,...) applies the global function
   INTEGRATE(x.tdata(:,ich1),x.F,...) to the specified channels ich1 of the
   mcsignal x.

   x          mcsignal object.
   ich1       Channels to integrate.  Default: all channels.
   y          mcsignal object.

   Use y = x.integrate{ich1}(...) to access this function.

   See also the global function INTEGRATE.
```

5.1.28 key

```
KEY   Extract one key value from a list of key options with unknown length.

   y = KEY(name,default,'KeyName','KeyValue',...) compares the 'KeyName's
   with 'name' and returns the following 'KeyValue'.  If no 'KeyName' matching
   'name' is found, 'default' is returned.

   name      Name of the argument to retrieve the value of.
   default   Default argument value.
   y         Argument value.

   See also VARARGIN.
```

5.1.29 labels

```
LABELS   Multi channel signal label per channel.
   y = LABELS(x) returns a cell array containing strings describing the label
   for each channel of the mcsignal x.

   x          mcsignal object.
   y          Cell array containing label describing strings, e.g. '12z'.

   Use y = x.labels to access this function.
```

5.1.30 mcsignal

```
MCSIGNAL   Multi channel time signal (unity time = 1 s) constructor.
   y = MCSIGNAL(s1,s2,...) assembles the mcsignals s1, s2,... to a single
   mcsignal.
   y = MCSIGNAL(ddffile,quantity,sifactor,labels) creates a mcsignal
   containing the data in the specified DasyLab DDF file.  Quantity
   defaults to 'volt' and sifactor defaults to 1.
```

```
   y = MCSIGNAL(tdmfile,quantity,sifactor,labels) creates a mcsignal
   containing the data in the specified LabVIEW TDM file.  Quantity
   defaults to 'volt' and sifactor defaults to 1.
   y = MCSIGNAL(wavfile,quantity,sifactor,labels) creates a mcsignal
   containing the data in the specified Test.Lab WAV file.  Quantity
   defaults to 'volt' and sifactor defaults to 1.
   y = MCSIGNAL(ascfile,F,quantity,sifactor,labels) creates a mcsignal
   containing the data in the specified ASCII file.  Sifactor defaults to 1.
   y = MCSIGNAL(x,F,quantity,sifactor,labels) creates a mcsignal
   containing the data in the specified matrix x.  Columns are interpreted
   as channels. Sifactor defaults to 1.

   s1,s2,...   mcsignal objects to assemble into 1 mcsignal.
   ddffile     DasyLab DDF file name.
   ascfile     Name of an ASCII file containing a multi channel time signal.
               Columns are interpreted as channels.
   x           Matrix containing a multi channel time signal.
               Columns are interpreted as channels.
   F           Sampling frequency, *NOT* the Nyquist frequency.
   quantity    Text string describing the data type per channel, e.g. 'volt',
               'acc', 'velo', 'disp', force'.  If not equal for all channels
               then provide a cell array containing different text strings.
   sifactor    Factor to multiply the stored signal data with as to obtain the
               signal in the SI units corresponding to the specified quantity.
               If not equal for all channels then provide a vector.
   labels      Text string describing the different channels. Defaults to
               the channel number

   See the example master files for info on the usage of the MCSIGNAL class.
```

5.1.31 md5

```
MD5   Generate MCSIGNAL signature using MD5 algorithm.
   y = MD5(x,accu) returns a 512-bit signature identifying all data stored in
   the mcsignal x: tdata, F, quantity, sifactor.  The tdata is taken into
   account with the specified accuracy: deviations smaller than
   MAX(tdata(:)) * accu are ignored.

   b = MD5(...,y) returns true if the signature of the mcsignal x equals y.

   x          mcsignal object.
   accu       Accuracy, default: 1e-8
   y          MD5 signature.
   b          Boolean.

   Use y = x.md5 to access this function.
```

5.1.32 n_

```
N_   Multi channel signal length.
   y = N_(x) returns the number of samples per channel for the mcsignal x.

   x          mcsignal object.
   y          Number of samples per channel.

   Use y = x.N to access this function.
```

5.1.33 nch

```
NCH   Multi channel signal number of channels.
   y = NCH(x) returns the number of channels for the mcsignal x.

   x          mcsignal object.
   y          Number of channels.

   Use y = x.nch to access this function.
```

5.1.34 noelec

```
NOELEC   Remove specified frequency components from a mcsignal.
   y = NOELEC(ich1,x,...) applies the global function
   NOELEC(x.tdata(:,ich1),x.F,...) to the specified channels ich1 of the
   mcsignal x.

   x          mcsignal object.
   ich1       Channels to remove frequency components from.  Default: all channels.
   y          mcsignal object.

   Use y = x.noelec{ich1}(...) to access this function.

   See also the global function NOELEC.
```

5.1.35 numel

```
NUMEL   Function called before subsref to determine the number of outputs.
   n = NUMEL(...) always returns 1.
```

5.1.36 plot

```
PLOT   Plot multi channel signal.
   PLOT(domain,ich1,x,filename,'KeyName','KeyValue',...) plots the time
   history [domain='t'], the frequency content (continuous Fourier transform)
   [domain='f'] or the one third octave band spectrum of the RMS value
   [domain='b'] of the selected channels ich1 of the mcsignal x.  Key options
   can be specified to fine tune the plot.  These options are interpreted by
```

```
the present function, and they are passed to the global function NEWFIG
which is used to open a new figure window.  The values of the key options
interpreted by the present function may be scalars / single strings, or
vectors / cell arrays of strings.  In the first case, the same value is
employed for all channels, while in the second case, values for all of the
mcsignal's channels (not only the selected channels ich1) have to be
provided.

domain       't' to plot time history, 'f' to plot frequency content, or 'b'
             to plot the one thirds octave band spectrum of the RMS value.
ich1         Channels to plot.  Default: all channels.
x            mcsignal object.
filename     File to save the plot to, using the global function SAVEFIG.
             Can be omitted if you do not want to save the plot.
'AddToPlot'  Command evaluated after all built in plotting routines, as to
             add elements to the figure, e.g. equal human response curves.
'AxesSize'   Size of the axes in centimeters.  See NEWFIG for more info.
'Blank'      Do not plot the curve, plot only the axes and the labels.
'Data'       Value is assigned to the local variable 'Data', which can be
             used by the 'AddToPlot' command.  If a cell array is provided,
             the contents of cell i is assigned to the local variable 'Data'
             when plotting channel i.
'ix'         First index of the time interval to plot.
             Default: 1
'jx'         Last index of the time interval to plot.
             Default: x.N
'Language'   Label language.
             'en' or 'nl'
             Default: 'en'
'Legend'     Default: no legend.  See also LEGEND.
'LegendLoc'  Legend location.  Default: 'Best'.  See also LEGEND.
'LineWidth'  Default: 0.5 for domain 't' and 'f', and 1.5 for domain 'b'.
'Margins'    Margins between the axes and the bounding box, in centimeters.
             See NEWFIG for more info.
'NXTick'     Number of XTicks (approximately, see the global function TICK).
             Default: auto.
'NYTick'     Number of YTicks (approximately, see the global function TICK).
             Default: auto.
'XLabel'     X-axis label.
             Default: defined using MCSIGNAL/LABEL
'YLabel'     Y-axis label.
             Default: defined using MCSIGNAL/LABEL
'XMin'       X-axis scaling: minimum value.
             Default: left side of the plotted curve.
'XMax'       X-axis scaling: maximum value.
             Default: right side of the plotted curve.
'YMin'       Y-axis scaling: minimum value.
             Default: defined as a function of the plotted curve Y(X):
             -max(abs(Y)) for domain 't', 0 for domain 'f' and min(Y) for
             domain 'b'.
```

```
   'YMax'       Y-axis scaling: maximum value.
                Default: defined as a function of the plotted curve Y(X):
                max(abs(Y)) for domain 't' and max(Y) for domains 'f' and 'b'.

   See also NEWFIG, SAVEFIG, MCSIGNAL/TPLOT, MCSIGNAL/FPLOT, MCSIGNAL/BPLOT.
```

5.1.37 quantity

```
QUANTITY   Multi channel signal data type per channel.
   y = QUANTITY(x) returns a cell array containing strings describing the data
   type for each channel of the mcsignal x.

   x          mcsignal object.
   y          Cell array containing data type describing strings, e.g. 'volt'.

   Use y = x.quantity to access this function.
```

5.1.38 resample

```
RESAMPLE   Resample a mcsignal.
   y = RESAMPLE(x,P) applies the global function
   RESAMPLE(x.tdata,P,x.F) to the channels of the mcsignal, thus setting
   the sampling frequency of x to P.
   y = RESAMPLE(x,P,Q,...) applies the global function
   RESAMPLE(x.tdata,P,Q,...) to the channels of the mcsignal x.

   x          mcsignal object.
   y          mcsignal object.

   Use y = x.resample(P) or x.resample(P,Q,...) to access this function.

   See also the global function RESAMPLE.
```

5.1.39 select

```
SELECT   Delete channels from a mcsignal.
   y = SELECT(ich1,x) deletes the channels ich0 from the mcsignal x.
   Only the channels ich1 are kept.

   x          mcsignal object.
   ich1       Channels to keep.
   y          mcsignal object.

   Use y = x.select{ich1}(...) to access this function.

   See also MCSIGNAL/DELETE.
```

5.1.40 setn_

```
SETN_   Set number of samples of mcsignal.
   y = SETN(x,N) truncates the mcsignal x if it is longer than N samples and
   pads is with zeros otherwise.

   x          mcsignal object.
   N          number of samples.
   y          mcsignal object.

   Use y = x.setN(N) to access this function.
```

5.1.41 setsifactor

```
SETSIFACTOR   Convert mcsignal units.
   y = SETSIFACTOR(ich1,x,sifactor) sets the SI factor (more info: see MCSIGNAL)
   of the selected channels to the specified value.  E.g. to convert an
   acceleration signal 'x' from [m/s^2] to [g], use
   y = SETSIFACTOR(ich1,x,9.81).

   x          mcsignal object.
   ich1       Channels to convert.  Default: all channels.
   sifactor   New SI-factor, e.g. 9.81 for the conversion from [m/s^2] to [g].
              If not equal for all channels then pass a vector.
   y          mcsignal object.

   Use y = x.setsifactor{ich1}(sifactor) to access this function.
```

5.1.42 sifactor

```
SIFACTOR   Multi channel signal SI-factor.
   y = SIFACTOR(x) returns the factors to multiply the stored signal data with
   as to obtain the signal in the SI units corresponding to the channel's
   quantity.

   x          mcsignal object.
   y          SI-factors.

   Use y = x.sifactor to access this function.
```

5.1.43 subsref

```
SUBSREF   Handle references to the methods of a mcsignal.
   SUBSREF(x,ref) is called by Matlab whenever one of the following commands is
   encountered: x(...), x.method{ich1}(...), x.method(...), etc.

   x(...) is conceptually equal to x.tdata(...).
```

```
   x.abcd{ich1}(...) results in the execution of the method
   abcd(x,ich1,ich0,...) where ich1 is a vector containing the channels
   specified by the user and ich0 is a vector containing the remaining channels.

   x.abcd(...) results in the execution of the method
   abcd(x,ich1,ich0,...) where ich1 is a vector containing all channels and ich0
   is an empty vector.

   To take into account the case insensitivity of Windows file names and
   therefore names of M-files, the called method is converted to an M-file name
   as follows: an underscore '_' is inserted right after every capital and
   every underscore.
```

5.1.44 t

```
T   Multi channel signal time vector.
   y = T(x) returns a vector containing the times corresponding to the samples
   stored in the mcsignal x.

   x          mcsignal object.
   y          Times [s].

   Use y = x.t to access this function.
```

5.1.45 t_

```
T_   Multi channel signal period.
   y = T_(x) returns the period of the mcsignal x.

   x          mcsignal object.
   y          Period [s].

   Use y = x.T to access this function.
```

5.1.46 tdata

```
TDATA   Multi channel signal time history.
   y = TDATA(x) returns the time history of the mcsignal x.

   x          mcsignal object.
   y          Time history.

   Use y = x.TDATA to access this function.
```

5.1.47 tplot

```
TPLOT   Plot multi channel time history.
   TPLOT(ich1,x,filename,'KeyName','KeyValue',...) executes the function
```

```
   PLOT('t',ich1,x,filename,'KeyName','KeyValue',...).

   Use x.tplot{ich1}(filename,'KeyName','KeyValue',...) to access this function.

   See also MCSIGNAL/PLOT.
```

5.1.48 trim

```
WINDOW   Restrict an mcsignal to a certain range of samples.
   y = WINDOW(x,s_l,s_u) selects the signals from sample s_l and s_u and

   x          mcsignal object.
   s_l        lower sample number
   s_u        upper sample number
   y          mcsignal object.
```

5.1.49 volt2x

```
VOLT2X   Convert multi channel signal from Volt to actual units.
   y = VOLT2XC(ich1,x,sensnr,...) applies the global function
   VOLT2X(x.tdata(:,ich1),S,...) to the specified channels ich1 of the
   mcsignal x.  The sensitivities S and the quantities to convert to result
   from the global function SENSITIVITY(sensnr).

   x          mcsignal object.
   ich1       Channels to convert.  Default: all channels.
   sensnr     Sensor numbers, specified in one of the files 'sensors-bwm.txt'
              and 'sensors-other.txt', which should be located in the Matlab
              path.
   y          mcsignal object.

   Use y = x.volt2x{ich1}(sensnr,...) to access this function.

   See also the global functions VOLT2X and SENSITIVITY.
```

5.1.50 volt2xa

```
VOLT2XA   Convert multi channel signal from Volt to actual units.
   y = VOLT2XA(x,...) applies the global function
   VOLT2X(x.tdata,...) to all channels of the mcsignal x.

   x          mcsignal object.
   sens       Sensitivities.  For accelerations: [V/(m/s^2)]
   amp        Amplification factors.  Can be in [dB].
   db         1 if amp is in [dB], 0 otherwise.
   quantity   Text string describing the data type per channel, e.g. 'volt',
              'acc', 'velo', 'disp', force'.
   label      Label defining the channel, e.g. '5z'
```

```
   y          mcsignal object.

   See also the local function VOLT2X and the global function VOLT2X.
```

5.1.51 weight

```
WEIGHT   Apply frequency weighting to a mcsignal.
   y = WEIGHT(ich1,x,H) multiplies the frequency content of the
   specified channels ich1 of the mcsignal x with the weighting function H.
   Use FFTFREQ to obtain the frequencies to use for the definition of H.

   x          mcsignal object.
   ich1       Channels to weight.  Default: all channels.
   H          Weighting function.
   y          mcsignal object.

   Use y = x.weight{ich1}(H) to access this function.

   See also MCSIGNAL/FFTFREQ.
```

5.1.52 window

```
WINDOW   Apply a time window to a mcsignal.
   y = WINDOW(ich1,x,w) multiplies the selected channels of the mcsignal x with
   the corresponding channels of the window function w.  If w only contains a
   single channel then this one is used for all selected channels of x.

   x          mcsignal object.
   ich1       Channels to apply the window to.  Default: all channels.
   w          Window function.  Must have x.N rows and 1 or x.nch columns.
   y          mcsignal object.

   Use y = x.window{ich1}(w) to access this function.

   See also MCSIGNAL/TRIM and the global functions DINTRIMDATA and BUTTERWINDOW.
```

5.2 Conversion

5.2.1 cell_

```
CELL_ Convert char to cell

   x = cell_(y)

   y       a char of size (m,n)
   x       a cell of size (m,1)

   Note    white spaces at the ends are trimmed
```

5.2.2 fe2ss

```
FE2SS Conversion of a finite element to a state-space model.

 [A,B,C,D] = fe2ss(K,M,Cv,Ts)
 [A,B,C,D] = fe2ss(K,M,Cv,Ts,DOF,uDOF,udDOF,uddDOF,iDOF)

 A,B,C,D   state-space matrices. If Ts=0, the description is in
           continuous-time. If Ts>0, the description is in discrete-time.
           In the discretization, a ZOH assumption is made.
 K,M,Cv    finite element stiffness, mass, and viscous damping matrices
 Ts        sampling time interval. If Ts = 0, the resulting description is
           in continuous-time.
 DOF       vector with degree of freedom numbers of K,M, and Cv (optional)
 uDOF      output displacement degrees of freedom numbers (optional)
 udDOF     output velocity degrees of freedom numbers (optional)
 uddDOF    output acceleration degrees of freedom numbers (optional)
 iDIF      input force degrees of freedom numbers (optional)
```

5.2.3 input2mcsignal

```
INPUT2MCSIGNAL This function converts measurement files to objects of the
   mcsignal class. The MCSIGNAL class combines the storage of the time
   history of a multi channel signal, the sampling frequency and the
   quantities and units of all channels with a set of methods (1) to
   determine the signal's metadata in all of its forms (sampling
   frequency, time period, time step, resolution in the frequency domain,
   Nyquist frequency,...), (2) to access the time history and the
   frequency content of the signal, and (3) to perform operations on the
   signal (integrate, decimate,...).

   x = input2mcsignal(file, ext, f_sample)

   file        a char containing in each row the name of the file that
               needs to be converted
   ext         a char containing in each row the corresponding file type
               ('mat', 'asc', 'ddf', 'msd', 'tdm', 'wav', 'f32')
   f_sample    sampling frequency (only needed when at least one of the
               extentions is equal to 'asc')
   x{1}, x{2}, ...
               the created mcsignal objects

   NOTE: When the extension equals 'msd', the different files are assumed
         to be measured simultaneously, and they are always combined into a
         single mcsignal object.
```

5.2.4 modal2ss

```
MODAL2SS   Compose a deterministic, stochastic, or combined
```

```
  deterministic-stochastic discrete-time state-space model from modal
  parameters. A Zero-Order-Hold (ZOH) discretization scheme is assumed.

 [Lambda,Bm,Phi,D,Km] = modal2ss(fud,xi,Phi,Ld,Km,Ts,types)

 [Lambda,Bm,Phi,D,Km] combined deterministic-stochastic state-space model
       in decoupled (i.e., modal) form. In the special case of a purely
       stochastic state-space model, Bm and D are empty. In the special
       case of a purely deterministic state-space model, Km is empty.
 fud   vector with undamped eigenfrequencies.
 xi    vector with modal damping ratios (units: % of critical).
 Phi   matrix with mode shapes.
 Ld    matrix with deterministic modal participation vectors.
 Km    vector with Kalman filter in modal form.
 Ts    samping period (units: s).
 types vector with output data types.
```

5.2.5 rmfd2ss

```
RMFD2SS Conversion of a Right Matrix Fraction Description model to a
   frequency-domain state-space model, i.e. a state-space model with
   complex matrices

   [A,B,C,D] = rmfd2ss(Ap,Bp)

   Ap, Bp    3D matrices containing the matrix polynomials of the rmfd
             description (3rd dimension: polynomial order)
   A,B,C,D   complex matrices of the equivalent state space description
```

5.2.6 rmfd2sysmat

```
 RMFD2SYSMAT convert structure of identified Right Matrix Fraction
   Description models to an equivalent structure of State Space models

   sysmat = rmfd2sysmat(rmfd)

   rmfd   structure of identified RMFD models
   sysmat structure of SS models
```

5.2.7 volt2x

```
VOLT2X   Convert data from Volts to actual units.

   [x,r] = VOLT2X(volt,sens,amp,db,direction)

   volt        Data in Volt.
   sens        Sensitivities.  For accelerations: [V/(m/s^2)].
   amp         Amplification factors.  Can be in [dB].
```

```
   db          1 if amp is in [dB], 0 otherwise.  Can be specified for all
               channels individually (1 * nch) or at once (1 * 1).
   direction   Defaults to +1 for every channel.  Specify -1 to revert a
               channel.  Can be specified for all channels
               individually (1 * nch) or at once (1 * 1).
   x           Data in actual units.  For accelerations: [m/s^2].
   r           Conversion rates per channel. For accelerations: [m/s^2/V].
```

5.3 Mathematics

5.3.1 Kronpr

```
KRONPR Calculation of the Kronecker product

   X = Kronpr(A,B)

   A,B     matrices for which the Kronecker product has to be calculated
   X       the Kronecker product of A and B
```

5.3.2 blckComp

```
BLCKCOMP Calculation of Block Companion matrix

   x = blckComp(A)

   x       Resulting Block Companion matrix
   A       3D-matrix comtaining the block colmns of the block Companion matrix
           (the third dimension corresponds to the block row position)
```

5.3.3 blckHank

```
BLCKHANK Calculation of Block Hankel matrix

   x = blckHank(C,R)

   x       Resulting Block Hankel matrix
   C       3D-matrix containing the first block column of the block Hankel matrix
           (the third dimension corresponds to the block row position)
   R       3D-matrix containing the last block row of the block Hankel matrix
           (the third dimension corresponds to the block column position)
```

5.3.4 blckToep

```
BLCKTOEP Calculation of Block Toeplitz matrix

   x = blckToep(C,R)
```

```
   x       Resulting Block Toeplitz matrix
   C       3D-matrix containing the first block column of the block Toeplitz matrix
           (the third dimension corresponds to the block row position)
   R       3D-matrix containing the first block row of the block Toeplitz matrix
           (the third dimension corresponds to the block column position)
```

5.3.5 blckVec

```
BLCKVEC Calculation of Block Vector

   x = blckVec(Y)

   x      Resulting Block Vector
   Y      3D-matrix containing the block elements of the block Vector (the
          third dimension corresponds to the block number)
```

5.3.6 cov_

```
COV_ Calculation of sample covariance matrix

   Y = cov_(X)

   X       an m x 1 x n matrix containing m variables and n observations
   Y       the sample covariance matrix of X
```

5.3.7 eiglr

```
EIGLR calculation of eigenvalues and left and right eigenvectors of a
   matrix

   [L,X,Y] = eiglr(A)

   L       diagonal matrix with eigenvalues
   X       corresponding right eigenvectors
   Y       corresponding left eigenvectors
   A       the matrix for which the eigenvalue problem needs to be solved

   one has: X*L/X = A and Y*conj(L)/Y = A'
```

5.3.8 key

```
KEY   Extract one key value from a list of key options with unknown length.

   y = KEY(name,default,'KeyName','KeyValue',...) compares the 'KeyName's
   with 'name' and returns the following 'KeyValue'.  If no 'KeyName' matching
   'name' is found, 'default' is returned.

   name      Name of the argument to retrieve the value of.
```

```
   default   Default argument value.
   y         Argument value.

   See also VARARGIN.
```

5.3.9 re_

```
RE_ Stacking the real part of a matrix on top of the imaginary part

   y = re_(x)

   x       complex matrix (may also be purely real or purely imaginary) of
           size m x n
   y       real matrix of size 2m x n
```

5.3.10 vec_

```
VEC_ Vectorization of a matrix

   x = vec_(X)

   X       a matrix of size m x n
   x       a vector of size mn x 1
```

5.3.11 vecuns

```
VECUNS unselect from vector

   y = vecuns(x,ind)

   x       vector from which some elements are unselected
   ind     the element numbers of x which have to be unselected
           (not the elements themselves)
   y       x without ind
```

5.4 Modal analysis

5.4.1 anpsd

```
ANPSD Calculate Averaged Normalized Power Spectral Density from Positive
   Power Spectral Density data, and related operational deflection shape
   (ODS) information.

 allmodes = anpsd(nonpar,quants)
 allmodes = anpsd(freqs,psdp,ref,quants)

 allmodes  a structure containing the operational deflection shape (ODS)
```

```
          information. Allmodes contains the following fields:
  allmodes.anpsd  averaged normalized power spectral density function.
                  size: 1 x n_f with n_f the number of frequency lines.
                  y is equal to the weighted mean of the Power Spectral
                  Densities of the reference outputs.
  allmodes.f      vector with corresponding frequencies [Hz]
  allmodes.m      matrix with operational deflection shapes in its
                  columns [m]. For a particular frequency, the
                  corresponding ODS is computed as the column containing
                  the largest element of the PSD matrix.
  allmodes.mpc    vector with modal phase colinearities of the ODSs [-]
  allmodes.mpd    vector with mean phase deviations of the ODSs []
  allmodes.wscheme    string containing mode schaling information. Equals to
                  'unit', since the ODSs have been scaled to unit modal
                  displacement.
  allmodes.quants cell containing the physical quantities of each ODS
                  component.
 nonpar structure containing a system description identified using
       nonparametric system identification.
 freqs frequency lines for which the PSDP has been estimated.
 psdp  positive power spectral density matrix. size: n_o x n_ref x n_f
       with n_ref the number of reference outputs
 ref   vector with reference output numbers
 quants      a cell containing the measured output quantities.

 Note: in order to avoid negative values of the estimated ANPSD (the so-called
       positive realness problem), the function returns the magnitude of
       the estimated ANPSD.
 Note: damping ratios are not computed, since the half-power bandwidth
       method for computing damping ratios is inaccurate.
```

5.4.2 cmif

```
CMIF Calculate the Complex Mode Indication Function (CMIF), also referred
   to as Frequency Domain Decomposition (FDD), and related modal
   information, from Frequency Response Function (FRF, input-output) or
   Positive Power Spectral Density (PSD+, output-only) data.

 allmodes = cmif(nonpar,quants)
 allmodes = cmif(freqs,frf,quants,loc,ampl)
 allmodes = cmif(freqs,psdp,quants)

 allmodes  a structure containing the modal information. Allmodes contains
           the following fields:
  allmodes.f      vector with frequency lines [Hz]
  allmodes.s      matrix with corresponding FRF/PSD singular values
                  (second dimension: frequency).
  allmodes.m      3D-matrix with corresponding mode shapes (2nd
                  dimension: singular value; 3rd dimension: frequency).
```

```
  allmodes.mpc    matrix with modal phase colinearities of the ODSs (1st
                  dimension: singular value; 2nd dimension: frequency).
  allmodes.mpd    vector with mean phase deviations of the ODSs [] (1st
                  dimension: singular value; 2nd dimension: frequency).
  allmodes.wscheme    string containing mode schaling information. Equals to
                  'unit' (unit modal displacement scaling).
  allmodes.ld     matrix with deterministic discrete-time modal
                  participation vectors in each column [-]
  allmodes.quants cell containing the physical quantities of each mode
                  shape component.
nonpar structure containing a system description identified using
      nonparametric system identification.
freqs frequency lines for which the PSDP has been estimated.
psdp  positive power spectral density matrix. size: n_o x n_ref x n_f
      with n_ref the number of reference outputs
quants a cell containing the measured output quantities.
loc   driving point locations (output numbering). If loc equals zero,
      the modes are scaled to unity.
ampl  driving point amplitude.

Note: the PSD is computed by taking the sum of the provided PSD+ function
      and its complex conjugate, for each frequency line.
Note: damping ratios are not computed, since the half-power bandwidth
      method for computing damping ratios is inaccurate.
```

5.4.3 drawmodes

```
DRAWMODES  Plot a mode in the current axes

  drawmodes(mode3,modenr,amp,complmode,plotcolour,linewid,anim_, ...
              gridfile,path_beam,path_slave,nodplot,xlims,ylims,zlims,...
              axeq,grid_,phase_,view_)

  mode3       structure array containing mode shape information (see
              function mode3D.m)
  modenr      number of mode to draw
  amp         amplification factor for the modal displacements
  complmode   equals to 1 if a complex mode must be drawn, otherwise 0
  plotcolour  colour that is used for the plot (e.g. 'k', 'b', ...)
  linewid     line width to use
  anim_       equals to 1 if the plot is used in an animation
  path_beam   path to beam or surface file
  path_slave  path to slave file (empty string if no slaving is required)
  nodplot     equals to 1 if node numbers must be drawn, otherwise 0
  xlims       limits of X axis. if 0, default values
  ylims       limits of Y axis. if 0, default values
  zlims       limits of Z axis. if 0, default values
  axeq        equals to 1 if the axes have equal length, otherwise 0
  grid_       equals to 1 if the grid needs to be plotted, otherwise 0
```

```
   phase_      phase to plot
   view_       view matrix
```

5.4.4 globmod

```
GLOBMOD combining modal parameters of different setups to one global set
   of modal characteristics

   [stabglobmodes,global_nodes,global_dirs] = ...
                                   globmod(stabmodescell,node_num,meas_dir)

   stabmodescell
       cell for which each entry contains the modal parameters identified for
       a single measurement setup. Such entry is a structure of the same
       form as allmodes, see stable_propmodpar5.
   stabglobmodes
       structure containing the combined modal parameters. It has the same
       form as allmodes, see stable_propmodpar5. The following fields,
       however, are not present: 'o', 'mtn', 'df', 'dxi', 'mac', 'dmtn',
       'stdf', 'stdxi', 'stdmr', 'stdmi', 'stdmmax', 'cov', and 'quants'.
   node_num, meas_dir
           three-dimensional matrices containing in each depth level the node
           numbers and measurement directions for each setup. When the
           number of outputs is different amongst different setups, the
           order of setups should be changed such that the setup with the
           largest number of outputs

   Remarks:
       (1) Unity modal mass scaling is possible if the mode shapes of
           all setups are scaled in this way. Otherwise, the global mode
           shapes are scaled to unit modal displacement.
       (2) If all setups contain exactly the same DOFs, the mode shapes
           are averaged.

   See also stable_propmodpar5
```

5.4.5 mac

```
 MAC Modal Assurance criterion (matrix) calculation

 z = mac(x)
 z = mac(x,y)

 x,y:  matrices containing mode shapes in each column
 z:    if y is not specified, z is the mac matrix of x. If y is specified,
       z contains the MAC values between the modes in x and the modes in y.
```

5.4.6 mdtime

```
MDTIME   Time-domain modal decomposition of a measured output-only signal.

 [ym,z] = mdtime(y,Lambda,Phi,Km)

 ym    3D matrix; first dimension: output number, second dimension: mode
       number, third dimension: time.
 y     matrix with measured outputs.
 [Lambda, Phi, Km]  stochastic state-space matrices in decoupled (i.e.,
       modal) form.

 See also modal2ss.
```

5.4.7 mod3D

```
MOD3D calculate 3D mode shapes

   mode3 = mod3D(mod_sel,node_num,meas_dir,grid)

   mode3       a structure array consisting of the grid of the 3D mode
               shapes (mode3.grid) and the x, y and z components of the
               different mode shapes (mode3.x, mode3.y and mode3.z).
   mod_sel     a matrix of measured mode shapes
   node_num    a vector with node numbers
   meas_dir    a matrix with the measurement directions
   grid        the measurement grid
```

5.4.8 modeselect

```
MODESELECT select modes from allmodes, that mach the stabilization criteria.

   stabmodes = modeselect(allmodes,df,dxi,dm,dampco,damplco,trnormco,
                  dmtn,maxfrstd,maxdmpstd,maxmodstd,mpclb,mpub,mpdub)

   [stabmodes,stabfdmodes,stabfmmodes,stabfmodes,unstabmodes,selectedmodes] = ...
               modeselect(allmodes,df,dxi,dm,dampco,damplco,trnormco,dmtn, ...
                          maxfrstd,maxdmpstd,maxmodstd,mpclb,mpub,mpdub);

   stabmodes       has the same structure as allmodes, but contains only the
                   stable modes
   stabfdmodes     contains the modes that have stable frequency and damping,
                   but no stable mode shape, for whith the damping ratio
                   lies between the upper and lower bounds and that have
                   the highest modal transfer norms
   stabfmmodes     contains the modes that have stable frequency and mode
                   shape, but no stable damping, for whith the damping
                   ratio lies between the upper and lower bounds and
                   that have the highest modal transfer norms
```

```
   stabfmodes      contains the modes that have stable frequency, but no
                   stable mode shape or damping, for whith the damping
                   ratio lies between the upper and lower bounds and
                   that have the highest modal transfer norms
   unstabmodes     contains the unstable modes
   selectedmodes   contains the modes that have stable frequency,
                   for whith the damping ratio lies between the upper and
                   lower bounds and that have the highest modal transfer
                   norms
   allmodes        a structure containing all the modes (see stable_propmodpar5
                   for the structure of allmodes)
   df              frequency stabilization criterium [%]
   dxi             damping ratio stabilization criterium [%]
   dm              mode shape stabilization criterium [%]
   dampco          damping ratio upper bound [%]
   damplco         damping ratio lower bound [%]
   trnormco        if, for a certain model order, the modes are ordered in
                   descending modal transfer norm, only the first trnormco
                   modes are stable
   dmtn            modal transfer norm stabilization criterium [%]
   maxfrstd        upper bound on the standard deviation of the frequency
   maxdmpstd       upper bound on the standard deviation of the damping
                   ratio
   maxmodstd       upper bound on the standard deviation on any real or
                   imaginary component of the mode shape.
   mpclb           lower bound on the modal phase collinearity
   mpub            upper bound on the mean phase []
   mpdub           upper bound on the mean phase deviation []
```

5.4.9 modfind

```
MODFIND Find for a certain order the mode number in allmodes that is closest to
   a given frequency.

   modenr      the resulting mode number of allmodes
   yes         a value equal to 1 if the mode has not been selected yet,
               otherwise it is equal to 0
   selnrs_new  a vector containing all selected mode numbers
   allmodes    a structure containing all the modes (see stable_propmodpar5
               for the structure of allmodes)
   frequency   the frequency that is sought for
   order       the model order that is sought for
   selnrs_old  a vector containing all previously selected mode numbers
```

5.4.10 mp

```
MP Mean phase calculation, according to the Total Least Squares approach.

    y = mp(Psi)
```

```
[y,X] = mp(Psi)

Psi    matrix containing a mode shape in each column
y      vector containing the mean phases of the modes in  Psi (in degrees)
X      2-column matrix containing, in each row j, coordinates in the
       complex plane such that y(j) = atan(X(j,2)/X(j,1)).

   See also MPD, MPHC.
```

5.4.11 mpd

```
MPD Mean Phase Deviation, according to the Total Least Squares approach.

y = mpd(Psi)
y = mpd(Psi,weight)

Psi    matrix containing a mode shape in each column
y      vector containing the mean phase deviations of the modes in Psi (in
       degrees)
weight weighting factor. Valid values are 'abs' and 'none'. When 'abs' is
       selected, the absolute values of the mode shape components are used
       as weighting factors. If 'none' is selected, all weighting factors
       equal 1. Defaults to 'abs'. Note that if all weights equal zero
       (e.g., when the MPD of a zero vector is computed), y is set to
       zero.

   See also MP, MPHC.
```

5.4.12 mphc

```
MPHC Modal Phase Collinearity. The function calculates the modal phase
   collinearity of each mode in Psi according to pp 3-4 of
   R.S. Pappa, K.B. Elliot and A. Schenk: A consistent-mode indicator for
   the eigensystem realization algorithm. Technical report TM-107607, NASA,
   Hampton, VA, 1992

y = MPHC(Psi)

Psi    matrix containing a mode shape in each column
y      vector containing the modal phase collinearities of the modes in Psi

   See also MP, MPD.
```

5.4.13 mscf

```
 MSCF Modal scale factor (matrix) calculation

 z = mscf(x,y)
```

```
 x,y:  matrices containing mode shapes in each column
 z:    matrix containing the modal scale factors between the columns of x
       and the columns of y
```

5.4.14 npmodselect

```
 NPMODSELECT   Select modes after nonparametric system identification.

 stabmodes = npmodselect(allmodes,freqs)
 stabmodes = npmodselect(allmodes,freqs,svnums)

 allmodes  nonparametric allmodes structure (structure: see anpsd.m or
           cmif.m).
 freqs     vector containing selected frequency lines
 stabmodes structure containing the same fields as allmodes, but only at
           the selected frequency lines.
 svals     vector containing the singular value numbers at the selected
           frequency lines (needed for CMIF/FDD only).
```

5.4.15 propmodpar5

```
PROPMODPAR5   Modal parameters of a continuous-time or discrete-time state
   space model. If the model is in discrete-time, a zero-order-hold (ZOH)
   discretization assumption is used.

   [fud,xi,Phi,quants,Ld,Km,Gm,refquants] = propmodpar5(A,B,C,D,...
                              Q,R,S,G,Lambda0,Ts,quants,loc,ampl,refs)

   Phi     matrix containing in each column a mode shape (scaled to the mass
           matrix if the input location is specified), in displacement
           units.
   fud     row vector with the corresponding undamped frequencies
   xi      row vector with the corresponding damping ratios in [%]
   Ld      matrix containing in each column the corresponding
           discrete-time deterministic modal participation vector
   Km      matrix containing in each column the corresponding modal Kalman
           filter vector (only computed when the Control Systems Toolbox
           is available)
   Gm      matrix containing in each column the corresponding
           discrete-time stochastic participation vector; although the
           elements of Phi are re-scaled to displacement units, the
           elements of Gm are not; this s
   quants  cell containing the output quantities. Valid arguments are
           'acc' (accelerations), 'velo' (velocities), 'disp'
           (displacements), and 'strain' (strains). Since the output
           quantities are redefined to modal displacements or modal strains,
    quants is both an input and an output argument.
   refquants  cell containing the reference output quantities.
```

```
  A,B,C,D,Q,R,S,G,Lambda0 state-space system matrices. For a
          deterministic state-space description, obtained from an EMA test,
          only A,B,C,D are provided, and the others are empty. For a
          stochastic state-space description, obtained from an OMA test,
          either A,C,Q,R,S or A,C,G,Lambda0 are provided; the others are
          empty. For a combined deterministic-stochastic state-space
          description, obtained from an OMAX test, either A,B,C,D,Q,R,S
          or A,B,C,D,G,Lamda0 are provided; the others are empty.
  loc     driving point locations (output numbering). If loc equals zero,
          the modes are scaled to unity.
  ampl    driving point amplitude.
  refs    reference output numbers
  Ts      sampling period. If the state-space model is in
          continuous-time, Ts equals 0.

 Notes: (1) Since Phi is in displacement units, an FRF that is synthesized
            from the modal parameters obtained with propmodpar5 is a
            force-displacement FRF.
        (2) Since both Phi and Gm are rescaled to displacement units, a
            PSD(+) or output correlation sequence synthesized from these
            are in displacement-displacement units.
```

5.4.16 ssmodparvar

```
 SSMODPARVAR Calculate modal characteristics and their covariances from an
   identified state-space model

 [f,xi,Phi,covfxp,trnorm] = ssmodparvar(A,C,dt,ACcov,G,units)

 f       vector of eigenfrequencies
 xi      vector of corresponding damping ratios
 phi     matrix of mode shapes (each column contains a mode shape,
         normalized to unit maximum displacement or normalized to 1 at the
         corresponding value in units)
 units   vector with output numbers that are normalized to 1 (each entry
         corresponds to a mode) (optional). If unit is a scalar, the same output
         number is used for all modes.
 covfxp  covariance matrix between the eigefrequency, damping ratio and
         mode shape of a single mode. syntax:
         covfxp(:,:,k) = Cov([f(k);xi(k);re_(phi(:,k))])
         where Cov is a function that takes the covariance of a vector
 quants  cell containing the output quantities. Valid arguments are
         'acc' (accelerations), 'velo' (velocities), 'disp'
         (displacements), and 'strain' (strains). Since the output
         quantities are redefined to mode shape quantities in
         ssmodparvar, quants is both an input and an output argument.
 trnorm  vector with modal transfer infinity-norm values
 A,C,G   identified system matrices
 ACcov   covariance matrix of the identified matrices. This can be the full
```

```
        covariance matrix Cov([vec_(A);vec_(C)]), or, when the system
        description is in decoupled (or modal) form, ACcov can be a cell
        containing the covariance matrices for each decoupled system.
 dt     the sampling period
```

5.4.17 stable_propmodpar5

```
STABLE_PROPMODPAR5  Computing all the modes of a sequence of discrete-time
   state-space models

   allmodes = stable_propmodpar5(sysmat,Ts,chan_outin,ampl_outin,quants,covind)

   allmodes    a structure containing the mode information
   sysmat      structure containing all identified system matrices
   Ts          sampling period
   chan_outin  vector containing, for each input, the corresponding output
               DOF channel. If an input corresponds to no output DOF, the
               value equals zero.
   ampl_outin  vector containing, for each input, the weighting factor
               needed for unity modal mass weighting
   quants      a cell containing the measured output quantities.
   covind      optional boolean indicating whether or not the complete
               covariance matrices of the mode shapes are required. If
               covind == 1, then allmodes.cov is not empty. Defaults to 0.
   normunit    optional scalar containing the mode shape output number
               whos modal displacements are to be normalized to one. Is
               only effective when sysmat contains covariance matrices.

 The structure allmodes has the following fields:
   allmodes.f      vector with eigenfrequencies [Hz]
   allmodes.o      vector with model orders [-]
   allmodes.xi     vector with damping ratios [%]
   allmodes.m      matrix with mode shapes in each column [m]
   allmodes.ld     matrix with deterministic discrete-time modal
                   participation vectors in each column [-]
   allmodes.km     matrix with modal Kalman filter vectors in each column
                   (not empty when the covariance matrices of the process
                   and measurement noise are available in sysmat) [-]
   allmodes.gm     matrix with stochastic modal participation vectors in each column
                   (not empty when the covariance matrices of the
                   stochastic states at time k+1 and the stochastic
                   outputs at time k are available in sysmat) [-]
   allmodes.trinfd vector with deterministic modal transfer infinity-norms
                   [abs(velocity/force)]
   allmodes.trinfs vector with stochastic modal transfer infinity-norms
                   [velocity]
   allmodes.mpc    vector with modal phase colinearities [-]
   allmodes.mp     vector with mean phases []
   allmodes.mpd    vector with mean phase deviations []
```

```
allmodes.dlambdac     vector with minimal relative continuous-time
                eigenvalue distance to a mode of the nearest lower model order [%]
allmodes.df     vector with relative eigenfrequency difference w.r.t.
                the same mode of the nearest lower model order as in
                allmodes.dlambdac [%]
allmodes.dxi    vector with relative damping ratio difference w.r.t.
                the same mode of the nearest lower model order as in
                allmodes.dlambdac [%]
allmodes.Dxi    vector with absolute damping ratio difference w.r.t.
                the same mode of the nearest lower model order as in
                allmodes.dlambdac [%]
allmodes.mac    MAC value between the current mode and the same mode
                of the nearest lower model order as in allmodes.df [%]
allmodes.dtrinfd  vector with relative deterministic modal transfer
                infinity norm difference w.r.t. the same mode of the
                nearest lower model order as in allmodes.dlambdac [%]
allmodes.dtrinfs  idem, but for stochastic infinity-norm
allmodes.stdf   vector with standard deviations of the eigenfrequencies [Hz]
allmodes.stdxi  vector with standard deviations of the damping ratios [%]
allmodes.stdmr  matrix with standard deviations of the real part of the
                mode shapes
allmodes.stdmi  matrix with standard deviations of the imaginary part
                of the mode shapes
allmodes.stdmmax    vector with the maximal standard deviations of the
                mode shapes
allmodes.cov    3D matrix containing in allmodes.cov(:,:,k) the
                covariance matrix of
                   [allmodes.f(k);
                    allmodes.xi(k);
                    real(allmodes.m(:,k));
                    imag(allmodes.m(:,k))].
allmodes.wscheme    string containing mode schaling information. If equal to
                'unit', the mode shapes have been scaled to unit modal
                displacement. If equal to 'mass', the mode shapes have been
                scaled to unity modal mass.
allmodes.quants cell containing the physical units of each mode
                shape component.
allmodes.refquants  cell containing the physical units of each
                reference output
allmodes.conj   boolean indicating whether, for a particular mode,
                another mode with complex conjugate eigenvalue has also
                been identified. If so, only the mode with positive
                eigenfrequency is listed in allmodes

Note: if some quantities (e.g., standard deviations) are not available,
the corresponding parts of allmodes are empty.
```

5.4.18 stabpick

```
STABPICK Pick selected modes out of a structure named allmodes

   stabmodes = stabpick(allmodes,selnrs,crit)

   stabmodes   a structure containing the selected modes (see
               stable_propmodpar.m for the structure of stabmodes)
   allmodes    a structure containing all the modes (see
               stable_propmodpar.m for the structure of allmodes)
   selnrs      a vector containing the selected mode numbers
   crit        the criterium according to which the poles are sorted.
               Defaults to 'ascend'. Other possibility: 'trnorm'
               (in decending order of modal transfer norm)
```

5.4.19 stabplot

```
STABPLOT plot stabilization diagram

   stabplot(allmodes,staballmodes,stabfdmodes,stabfmmodes,stabfmodes, ...
               unstabmodes,stabonly, plotall,colorind,f_low,f_high, ...
               freqscale,psdpsum,frfsum)

   allmodes        a structure containing all the modes (see stable_propmodpar.m
                   for the structure of allmodes)
   staballmodes    has the same structure as allmodes, but contains only
                   the stable modes
   stabfdmodes     contains the modes that have stable frequency and damping,
                   for which the damping ratio lies between the upper and
                   lower bounds, that have the highest modal transfer
                   norms, and for which the modal phase colinearity is higher
                   than the threshold and the mean phase deviation lower
                   than the threshold.
   stabfmmodes     contains the modes that have stable frequency and mode
                   shape, for whith the damping ratio lies between the
                   upper and lower bounds, that have the highest modal
                   transfer norms, and for which the modal phase
                   colinearity is higher than the threshold and the mean
                   phase deviation lower than the threshold.
   stabfmodes      contains the modes that have stable frequency, for which
                   the damping ratio lies between the upper and lower
                   bounds, that have the highest modal transfer norms, and
                   for which the modal phase colinearity is higher than
                   the threshold and the mean phase deviation lower than
                   the threshold.
   stabonly        integer equal to 1 if only the stable modes need to be
                   plotted
   plotall         integer equal to 1 if also the unstable modes need to
                   be plotted
```

```
colorind        integer equal to 1 if the plot must be in color,
                otherwhise it is in greyscale
f_low           frequency lower bound
f_high          frequency higher bound
freqscale       frequency scale (optional)
psdpsum         sum of power spectral densities (optional)
frfsum          sum of frequency response functions (optional)
```

5.5 System identification

5.5.1 H1estimate

```
H1ESTIMATE     FRF estimation using the H1 estimate

   [h1frf,freqscale] = H1estimate(predat,Nblocks,windowtype,...
                                          inputs,outputs,fmaxstr,pgbar)
   [h1frf, h1cov, freqscale] = H1estimate(predat,Nblocks,windowtype,...
                                          inputs,outputs,fmaxstr,pgbar)

   h1frf       frf estimate using H1 method. row: outputs, columns:
               frequencies, depth: inputs
   h1cov       3D matrix of covariances on the H1 estimate. row: outputs,
               columns: frequencies, depth: inputs
   freqscale   frequency scale
   predat      mcsignal object
   Nblocks     number of blocks to be used in averaging procedure
   windowtype  window type to be used in averaging procedure. Valid
               arguments are 'rect' for a rectangular window and 'hann'
               for a Hanning window
   inputs      vector with channel input numbers
   outputs     vector with channel output numbers
   fmaxstr     string indicating whether the maximal frequency in h1frf
               should be the Nyquist frequency ('fn'), or the sampling
               frequency ('fs').
   pgbar       boolean indicating whether a progressbar should be shown
               (1) or not (0).
```

5.5.2 PSDpos_corr

```
PSDPOS_CORR Estimation of Positive Power Spectral Densities (PSD+s) via
   the correlogram method

   [psdp,freqscale,cor,timescale] = PSDpos_corr(predat,Nlags,outputs,...
                                                refs,fmaxstr,dart)

   psdp        positive power spectrum estimate. row: outputs, columns:
               frequencies, depth: inputs
   cor         corresponding correlations.
```

```
   freqscale   frequency scale
   predat      mcsignal object
   Nlags       number of time lags to be used (zero lag included). Nlags
               equals the total number of (positive) frequency points
   outputs     vector with channel output numbers
   refs        vector with reference channel output numbers
   fmaxstr     string indicating whether the maximal frequency in psdp
               should be the Nyquist frequency ('fn'), or the sampling
               frequency ('fs') (optional, defaults to 'fn')
   dart        artificial damping (in % of critical) (optional, defauts to 0)
```

5.5.3 PSDpos_per

```
PSDPOS_PER Estimation of Positive Power Spectral Densities (PSD+s) using
   the periodogram approach

   [psdp,freqscale] = PSDpos_per(predat,Nblocks,outputs,refs,m,fmaxstr)
   [psdp,freqscale,cor,timescale] = ...
                      PSDpos_per(predat,Nblocks,outputs,refs,m,fmaxstr)
   [psdp,freqscale,cor,timescale,psdpcov] = ...
                      PSDpos_per(predat,Nblocks,outputs,refs,m,fmaxstr)

   psdp        positive power spectrum estimate. row: outputs, columns:
               frequencies, depth: inputs
   psdpcov     3D matrix of covariances on psdp. row: outputs, columns:
               reference outputs, depth: frequencies
   freqscale   frequency scale
   timescale   time scale
   cor         3D matrix of correlation functions (without applying the
               time window). row: outputs, columns: reference outputs,
               depth: time steps
   predat      mcsignal object
   Nblocks     number of blocks to be used in averaging procedure
   outputs     vector with channel output numbers
   refs        vector with reference channel output numbers
   m           time window parameter; if N is the total number of samples
               and Nw is the length of the time window, m = N/Nw
   fmaxstr     string indicating whether the maximal frequency in psdp
               should be the Nyquist frequency ('fn'), or the sampling
               frequency ('fs') (optional, defaults to 'fn')
```

5.5.4 RMFDcalc

```
RMFDCALC Calculation of Right Matrix-Fraction Description Polynomial.
   This function is used after a non-parametric preprocessing step during
   frequency-domain system identification.

   rmfd = rmfdcalc(H_meas,method,orders,chanselout,chanselin,...
                                              refs,freqscale,f_low,f_high)
```

```
   rmfd        a structure array, containing the A and B matrix polynomial
               coefficients for the given system orders
   H_meas      a 3D array, containing FRF, PSD+ or combined FRF/PSD+
               matrices for each frequency. The 3rd dimension is the
               frequency dimension.
   method      describes the system identification method to be used. If
               the Poly-reference Least Squares Complex Frequency Domain
               method has to be used, 'method' equals 'pLSCF'.
   chanselout  a vector, containing the output channels selected for
               identification
   chanselin   a vector, containing the input channels selected for
               identification
   refs        a vector, containing the reference channels selected for
               identification
   freqscale   the vector with frequencies, corresponding to the 3rd dimension
               of H_meas. f goes from 0Hz up to the sampling frequency.
   f_low       lower bound of the frequency range of interest
   f_up        upper bound of the frequency range of interest
```

5.5.5 csi_data

```
CSI_DATA  Data-driven Reference-based Combined Deterministic-Stochastic
   Subspace Identification

   invar = csi_data(F,Y,ii,ref)
   [A,B,C,D,Q,R,S] = csi_data(invar,n)

   F       the input data
   Y       the output data
   ii      half the number of block rows in the Hankel matrix
   invar   a structure which contains the results after QR and SVD
   n       the order of the identified system
   A,B,C,D the system matrices
   Q,R,S   the noise covariance matrices
```

5.5.6 identsel

```
IDENTSEL Select output and input node numbers, measurement directions,
   and quantities after system identification

   [node_numout,meas_dirout,quants_out,chan_outin,ampl_outin,refs] = ...
               identsel(node_num,meas_dir,quants,chanselout,chanselin,refers)

   node_numout vector with the output DOF node numbers
   meas_dirout matrix with output DOF measurement directions. Rows: DOFs,
               columns: direction angles
   quants_out  list of output DOF quantities (cell)
   chan_outin  vector containing the output DOF number corresponding to an
```

```
              input DOF. If an input has not the quantity 'force' or it
              does not correspond to exactly one output DOF, its value in
              chan_outin equals 0.
   ampl_outin vector containing the direction info of the corresponding
              element in chan_outin. Equals 1 if the output and input DOF
              have the same direction, otherwise -1. It can also have an
              intermediate value equal to the cosine of the angle between
              both directions.
   refs       output numbers (NOT channels) corresponding to the
              reference channels (optional)
   node_num   vector with all the node numbers
   meas_dir   matrix with all measurement directions. Rows: DOFs,
              columns: direction angles
   quants     string list of all quantities
   chanselout vector containing the output channels
   chanselin  vector containing the input channels
   refers     reference output channels (only required if refs is asked)
```

5.5.7 makenonpar

```
MAKENONPAR Make a structure containing a nonparametric system description

 nonpar = makenonpar(freqscale,realscale,frf,frfcov,psdp,psdpcov,inputs,outputs,refs)

 freqscale vector with discrete frequency lines where the
           nonparametric system description is known
 frf       estimated frequency response function data (first dimension: outputs,
           second dimension: inputs, third dimension: frequency)
 frfcov    covariance matrix of the estimated frf data. frfcov(:,:,k)
           contains the covariance matrix of vec_(frf(:,:,k))
 psdp      estimated positive power spectral density data (first dimension: outputs,
           second dimension: reference outputs, third dimension: frequency)
 psdpcov   covariance matrix of the estimated psdp data. psdpcov(:,:,k)
           contains the covariance matrix of vec_(psdp(:,:,k))
 inputs    input numbers
 outputs   output numbers
 refs      reference output numbers
```

5.5.8 modalmr

```
MODALMR    Modal model reduction of a state-space model

   [Ar,Cr] = modalmr(A,C,poles)
   [Ar,Br,Cr,Dr] = modalmr(A,B,C,D,poles)
   [Ar,Cr,Qr,Rr,Sr] = modalmr(A,C,Q,R,S,poles)
   [Ar,Br,Cr,D,Qr,R,Sr] = modalmr(A,B,C,D,Q,R,S,poles)

   A,B,C,D     system matrices in the original state-space description
   Q,R,S       noise covariance matrices in the original state-space
```

```
            description
   poles    a vector with the poles which are RETAINED. All system
            poles that are not in this matrix will be eliminated
   Ar,Br,Cr,Dr system matrices in the reduced state-space description
   Qr,Rr,Sr noise covariance matrices in the reduced state-space
            description
```

5.5.9 pLSCF4

```
pLSCF4 Poly-Reference Least Squares Complex Frequency Domain system
   (p-LSCF) identification. No frequency weighting is applied.

   [A,B] = pLSCF4(H,n,f,f_low,f_up)

   A, B    the right matrix fraction description matrices resulting from
           the identification. The third dimension corresponds to the
           exponent of the z minus 1.
   H       the measured transfer function matrix at different
           frequencies (3rd dimension). H may also contain positive
           power spectral densities (PSD+s) or a combination of FRFs and
           PSD+s.
   n       the required system order
   f       the vector with frequencies, corresponding to the 3rd dimension
           of H. f goes from 0Hz up to the sampling frequency.
   f_low   lower bound of the frequency range of interest
   f_up    upper bound of the frequency range of interest
```

5.5.10 ss2mss

```
SS2MSS     Conversion of a state-space description into its modal form

   [Lambda, Cm] = ss2mss(A, C)
   [Lambda, Bm, Cm, D] = ss2mss(A, B, C, D)
   [Lambda, Cm,  Qm, R, Sm] = ss2mss(A, C, Q, R, S)
   [Lambda, Bm, Cm, D, Qm, R, Sm] = ss2mss(A, B, C, D, Q, R, S)

   Lambda,Bm,Cm,Dm     the state-space description in modal form. Lambda
                       is a diagonal matrix.
   Qm,R,Sm             the state-space noise covariance matrices in modal
                       form.
   A,B,C,D             the original state-space matrices
   Q,R,S               the original state-space noise covariance matrices.
```

5.5.11 ssi_data3

```
SSI_DATA3  Data-driven Reference-based Stochastic Subspace Identification

   invar = ssi_data3(Y,ii,ref,algorithm)
```

```
  [A,C,Q,R,S] = ssi_data3(invar,n)

  Y           the output data
  ii          half the number of block rows in the Hankel matrix
  ref         column matrix containing the reference sensors
  algorithm   optional string containing the desired algorithm:
              'PC','UPC' or 'CVA'. Default: 'UPC'
  invar       results after QR and SVD.
  n           the order of the identified system
  A,C         the identified state-space matrices
  Q,R,S       the identified state-space covariance matrices
```

5.5.12 ssicov

```
SSICOV calculates the system matrices A, C, Gref and Lambda0ref from
  measured outputs y only, using reference-based stochastic subspace
  idenficiation (SSI-cov/ref) [1]. The computation is done in two
  commands; the first command should only be called once when several
  system descriptions, with different model orders, are identified from
  the same data, e.g. for constructing a stabilization diagram. The
  covariance matrix of the estimation errors can be optionally computed,
  according to the algorithm of [2,3]. The implementation of this
  covariance estimation procedure is much more memory efficient when
  compared to previous editions of MACEC, but this procedure may require
  most of the CPU time needed to run SSICOV.

invar = ssicov(Y,ii,refs)
invar = ssicov(Y,ii,refs,Nb)
invar = ssicov(invar,n_max)
[A,G,C,Lambda02] = ssicov(invar,n)
[A,G,C,Lambda02,covAC] = ssicov(invar,n)
[Lambda,Gm,Phi,Lambda02,covmod] = ssicov(invar,n,'decoupled')

Y     Matrix containing the output vectors as its columns
ii    Number of block rows in the block Hankel matrix
refs  vector containing the reference outputs (row number of Y)
      (optional; all outputs are reference outputs by default)
Nb    Number of blocks in the covariance estimation process (optional;
      equals to min(500,j) by default, with N the number of samples
      minus 2*ii). If N equals 1, the covariances of the system matrices
      are not estimated.
invar Structure containing intermediate identification results that are
      can be used at different model orders n.
A,G,C,Lambda02    System matrices of the estimated stochastic
      state-space description.
covAC             Complete statistical covariance matrix of the estimated
      stochastic state-space matrices A and C.
Lambda,Gm,Phi,Lambda02,covmod     System matrices of the estimated
      stochastic state-space description in decoupled format. 'covmod' is
```

```
      a structure containing the covariance matrices for each set of
      eigenparameters belonging to the same mode. This means that the
      information regarding the covariance between different modes is
      lost compared to covAC, yet the result is more memory efficient,
      especially for large model orders.

 [1] B. Peeters and G. De Roeck: Reference-based stochastic subspace
     identification for output-only modal analysis. Mechanical Systems and
     Signal Processing, 13(6):855-878, 1999.
 [2] E. Reynders, R. Pintelon, and G. De Roeck: Uncertainty bounds on modal
     parameters obtained from Stochastic Subspace Identification.
     Mechanical Systems and Signal Processing, 22(4):948-969, 2008.
 [3] M. Dhler and L. Mevel. Efficient multi-order uncertainty computation
     for stochastic subspace identification. Mechanical Systems and Signal
     Processing, 38(2):346-366, 2013.
```

5.5.13 sysmatcalc

```
SYSMATCALC Calculation of system matrices for increasing system orders.
   This function is used after a stochastic or a combined
   deterministic-stochastic subspace identification.

   sysmat = sysmatcalc(invar,method,chanselout,chanselin,refs,calcorders)
   sysmat = sysmatcalc(invar,method,chanselout,chanselin,refs,calcorders,coupling)

   sysmat      structure array, containing the A and C system matrices for
               every possible even system order
   invar       structure array, containing the R factor from the
               QR-decomposition, the U matrix and the singular values from
               the SVD and the new order of the channels (differs from the
               old order if the reference-based SSI was used)
   chanselout  vector, containing the output channels selected for
               identification
   chanselin  vector, containing the input channels selected for
               identification
   refs        vector, containing the output reference channels
   method      describes the system identification method used. If the
               ssi_data method was used, 'method' equals 'ssi_data'. If
               the ssi_cov method was used, 'method' equals 'ssi_cov'.
               If the csi_data method was used, 'method' equals 'csi_data'
   calcorders  vector, containing the model orders to be calculated
   coupling    optional string, indicating whether or not the full covariance
               matrix between all system matrix elements should be
               computed. This option is only effective when method =
               'ssi_cov'. When coupling = 'decoupled', the identified
               system matrices are in decopuled (modal) form, and the
               covariance between estimates belonging to different modes
               is not computed.
```

Chapter 6

Verification examples

In this chapter, full examples are provided where MACEC is used in batch mode. They have two purposes: (i) to demonstrate how the MACEC functions, which are explained in chapter 5, can be employed directly, without the GUI, and (ii) to verify the performance of MACEC by comparing the obtained results with analytical solutions or benchmark results reported in the literature. So far, only one example is provided. More examples will follow in future editions of this manual.

6.1 Example 1: 2DOF system

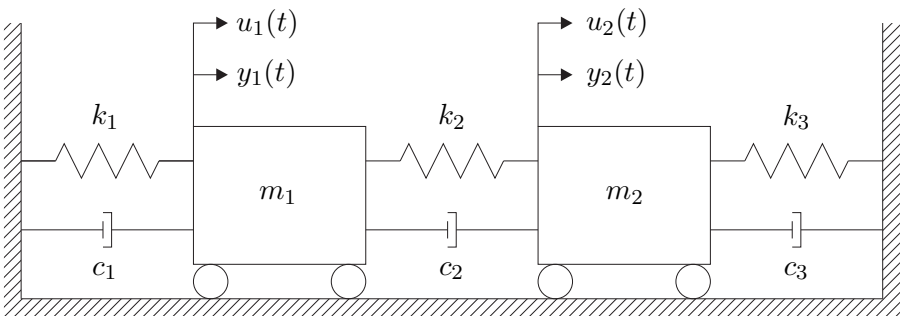


Figure 6.1: Verification example 1: 2DOF structure

In this example, a system with two degrees of freedom (DOFs) and proportional damping is simulated (fig. 6.1). It is used as an example in the classic textbook on modal testing by Heylen et al [5, section A.1.4.2]. The masses, dampers, and stiffnesses have the following values:

- $m_1 = m_2 = 2\text{kg}$
- $c_1 = c_3 = 3\frac{\text{N}}{\text{m/s}}$, $c_2 = 2\frac{\text{N}}{\text{m/s}}$
- $k_1 = k_3 = 4000\frac{\text{N}}{\text{m}}$, $k_2 = 2000\frac{\text{N}}{\text{m}}$.

With these values, the mass, stiffness, and damping matrices are computed as

$$\boldsymbol{M} = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \text{kg}$$

$$\boldsymbol{C} = \begin{bmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_2 + c_3 \end{bmatrix} = \begin{bmatrix} 5 & -2 \\ -2 & 5 \end{bmatrix} \frac{\mathrm{N}}{\mathrm{m/s}}$$

$$\boldsymbol{K} = \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 + k_3 \end{bmatrix} = \begin{bmatrix} 6000 & -2000 \\ -2000 & 6000 \end{bmatrix} \frac{\mathrm{N}}{\mathrm{m}}.$$

The system has two inputs $u_1(t)$ and $u_2(t)$, which are the horizontal forces applied to the masses m_1 and m_2, respectively. There are two outputs $y_1(t)$ and $y_2(t)$, which are the horizontal accelerations of the masses m_1 and m_2, respectively.

6.1.1 Example 1a: discretization

The purpose of this example is to check the influence of the Zero-Order-Hold (ZOH) assumption during discretization on the identified modal parameters. The assumption should only have an influence on the modal scaling factors, not on the eigenfrequencies, damping ratios, and unscaled mode shapes [10]. The Matlab source code of this example can be found in the file `Example1a.m` in the `examples` directory.

In MACEC, the system is defined and converted to a discrete-time state-space model with the following commands (note that a sampling frequency of 100Hz is chosen):

```
M = [2 0;0 2];                  % mass matrix
Cv = [5 -2;-2 5];               % damping matrix
K = [6000 -2000; -2000 6000];   % stiffness matrix
Ts = 1/100;                     % sampling period
[A,B,C,D] = fe2ss(K,M,Cv,Ts,[1.01 2.01],[],[],[1.01 2.01],[1.01 2.01]);
                                % discrete-time ss model
```

The modal parameters are computed as follows

```
quants = cell_(strvcat('acc','acc'));   % define the output quantities
[f_ud,xi,phi,quantsnew,L] = propmodpar5(A,B,C,D,[],[],[],[],[],Ts,quants,...
    [1 2],[1 1],[]);    % compute the modal parameters
```

The corresponding continuous-time modal parameters are obtained from

```
[Ac,Bc,Cc,Dc] = fe2ss(K,M,Cv,Ts,[1.01 2.01],[],[],[1.01 2.01],[1.01 2.01]);
                                        % continuous-time ss model
quants = cell_(strvcat('acc','acc'));   % define the output quantities
[f_ud_c,xi_c,phi_c,quantsnew,Lc] = propmodpar5(Ac,Bc,Cc,Dc,[],[],[],[],...
    [],0,quants,[1 2],[1 1],[]);    % compute the modal parameters
```

It can be verified that the eigenfrequencies and damping ratios obtained before and after discretization are equal. The mode shapes are also equal up to a scaling factor. The mass-normalized mode shapes, however, are not equal, due to the ZOH assumption. Table 6.1 below shows one of the mode shapes obtained for different sampling frequencies.

T_s	0	1/50	1/100	1/1000	1/100000
ϕ_{11}	$0.5001 + 0.0042i$	$0.4822 - 0.1054i$	$0.4963 - 0.0515i$	$0.5001 - 0.0014i$	$0.5001 + 0.0041i$
ϕ_{21}	$0.5001 + 0.0042i$	$0.4822 - 0.1054i$	$0.4963 - 0.0515i$	$0.5001 - 0.0014i$	$0.5001 + 0.0041i$

Table 6.1: Verification example 1a: mode shape of mode 1 at 7.1176Hz, obtained from discretization with a ZOH assumption for different sampling periods T_s.

6.1.2 Example 1b: pLSCF method

The purpose of this example is to check the performance of the deterministic pLSCF method for experimental modal analysis. Hereto, the system is discretized at a sampling frequency of 100Hz, and a sequence of inputs $\boldsymbol{u_k}$ and outputs $\boldsymbol{y_k}$ is simulated; the inputs are taken as random Gaussian sequences. The corresponding Matlab code reads

```
M = [2 0;0 2];                          % mass matrix
Cv = [5 -2;-2 5];                       % damping matrix
K = [6000 -2000; -2000 6000];           % stiffness matrix
Ts = 1/100;                             % sampling period
[A,B,C,D] = fe2ss(K,M,Cv,Ts,[1.01 2.01],[],[],[1.01 2.01],[1.01 2.01]);
                                        % discrete-time ss model
N = 8192;                               % number of samples
randn('state',0);                       % ensure repeatability
u = randn(2,N);                         % Gaussian random input
y = zeros(2,N); y(:,1) = D*u(:,1);      % initialization of outputs
x = zeros(4,N);
for par = 2:N
    x(:,par) = A*x(:,par-1) + B*u(:,par-1);
    y(:,par) = C*x(:,par) + D*u(:,par);
end
```

With the simulated data, a nonparametric FRF estimate is computed first using the H_1 approach [5], then a right-matrix fraction description (RMFD) model is computed using the pLSCF algorithm, the RMFD model is converted to a state-space model, and then the modal parameters are computed:

```
types = cell_(strvcat('force','force','acc','acc'));  % define data types
labels = cell_(strvcat('1F','2F','1A','2A'));         % define channel labels
predat = mcsignal([u' y'],1/Ts,types,1,labels);       % make an mcsignal object
                                                      % with the simulated data
node_num = [1;2;1;2];                                 % define node numbers
                                                      % for each channel
meas_dir = [0  0;0  0;0  0;0  0];                     % define measurement
                                                      % directions for each channel
[h1frf, h1cov, freqscale] = H1estimate(predat,4,'rect',[1  2],[3  4],'fs');
                                                      % estimate FRFs
rmfd = RMFDcalc(h1frf,'pLSCF',2,[3 4],[1  2],[],freqscale,0,50);
                                                      % estimate a right matrix
                                                      % fraction description of 3
                                                      % order 2 (exact order)
```

```
[node_num,meas_dir,quants,chan_outin,ampl_outin,refs] = ...
    identsel(node_num,meas_dir,predat.quantity',rmfd.chanselout,...
    rmfd.chanselin,rmfd.refs);
sysmat = rmfd2sysmat(rmfd);                          % convert to a state-space model
modes = stable_propmodpar5(sysmat,predat.dt,chan_outin,ampl_outin,quants);
                                                     % compute the modal parameters
```

Table 6.2 compares the identified modal parameters with the 'exact' values, obtained in section 6.1.1 after ZOH discretization with the same sampling frequency. The estimates are not exact, despite the fact that the data are purely deterministic. It can be verified that increasing the number of samples N increases the accuracy of the modal parameter estimates; decreasing this number decreases the accuracy.

	exact		deterministic pLSCF	
f_{ud}	7.1176	10.0658	7.1486	10.0337
ξ	1.68	2.77	1.29	2.57
$\boldsymbol{\phi}$	$0.4963 - 0.0515i$	$0.4930 - 0.0716i$	$0.4961 + 0.0572i$	$0.5014 - 0.0618i$
	$0.4963 - 0.0515i$	$-0.4930 + 0.0716i$	$0.4883 + 0.0417i$	$-0.4843 + 0.0803i$

Table 6.2: Verification example 1b: undamped eigenfrequencies f_{ud}, damping ratios ξ, and mode shapes $\boldsymbol{\phi}$: exact values vs. the estimates obtained with deterministic pLSCF.

Figure 6.2 compares the exact frequency response function (FRF), as found in [5, p.A1.30], with the H_1 estimate and the FRF that is computed with the identified RMFD description. The corresponding Matlab code can be found below

```
% compute FRFs
p = 1i*2*pi*freqscale;          % Laplace variables
nf = length(freqscale);         % number of frequency lines used in the plot
frf_ex = zeros(2,2,nf);         % initialize the matrix with exact FRFs
z = exp(Ts*p);                  % z variables
frf_rmfd = zeros(2,2,nf);       % initialize the matrix with FRFs estimated
                                % from the rmfd description
Ar2 = rmfd.A_2(:,:,3); Ar1 = rmfd.A_2(:,:,2); Ar0 = rmfd.A_2(:,:,1);
                                % define the 'A' matrices of the right matrix
                                % fraction description
Br2 = rmfd.B_2(:,:,3); Br1 = rmfd.B_2(:,:,2); Br0 = rmfd.B_2(:,:,1);
                                % define the 'B' matrices of the right matrix
                                % fraction description
for par = 1:nf,
    frf_ex(:,:,par) = [2*p(par)^2+5*p(par)+6000 p(par)+2000; ...
        p(par)+2000 2*p(par)^2+4*p(par)+6000]...
        /((2*p(par)^2+4*p(par)+6000)*(2*p(par)^2+5*p(par)+6000)-(p(par)+2000)^2);
    frf_ex(:,:,par) = frf_ex(:,:,par)*p(par)^2;
                                % compute the exact frf matrix
    frf_rmfd(:,:,par) = (Br2*z(par)^2+Br1*z(par)+Br0)/(Ar2*z(par)^2+Ar1*z(par)+Ar0);
                                % compute the synthesized frf matrix using the
```

```
                                % identified RMFD description
end
frf_ex(:,:,end/2+1:end) = conj(frf_ex(:,:,end/2:-1:1));

% create figure
newfig([10,12])
subplot(2,1,1),
semilogy(freqscale,abs(squeeze(frf_ex(1,1,:))),...
    freqscale,abs(squeeze(h1frf(1,1,:))),freqscale,abs(squeeze(frf_rmfd(1,1,:))))
xlabel('frequency [Hz]'), ylabel('abs(FRF) [m/N]'), xlim([0 50]),
legend('exact','H1','pLSCF','location','southeast')
title('Exact and estimated FRFs','fontsize',14)
subplot(2,1,2)
plot(freqscale,angle(squeeze(frf_ex(1,1,:))),...
    freqscale, angle(squeeze(h1frf(1,1,:))),...
    freqscale,angle(squeeze(frf_rmfd(1,1,:))))
xlabel('frequency [Hz]'), ylabel('angle(FRF) [rad]'), xlim([0 50]),
legend('exact','H1','pLSCF','location','southeast')
```

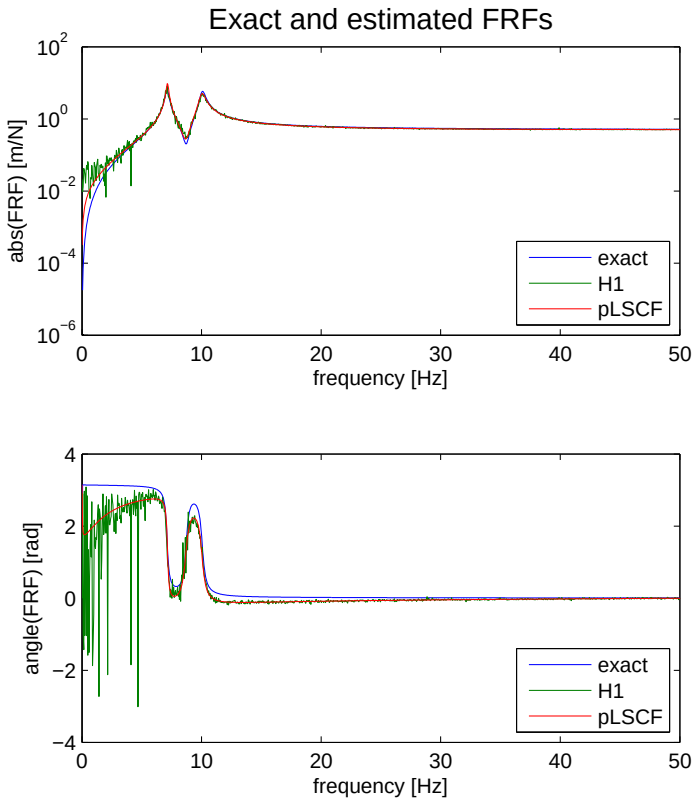


Figure 6.2: Verification example 1b: Element $(1,1)$ of the FRF matrix: exact values (blue), H_1 estimate (green), and the estimate obtained with the identified RMFD description using deterministic pLSCF (red).

6.1.3 Example 1c: modal time-domain response decomposition using SSI-data

The purpose of this example is to demonstrate how the decomposition of a measured response, using a stochastic state-space model identified from output only data, can be implemented in MACEC. For the theoretical background of the decomposition, we refer to [7, sec. 7.3].

First, simulate the response in exactly the same way as in section 6.1.2. Then, identify a stochastic state-space model using the SSI-data method:

```
ii = 10;                                    % half the number of block rows
refs = [1 2];                               % reference sensors
n = 4;                                      % system order
types = cell_(strvcat('acc','acc'));        % define data types
labels = cell_(strvcat('1A','2A'));         % define channel labels
predat = mcsignal(y',1/Ts,types,1,labels);  % make an mcsignal object
node_num = [1;2];                           % define node numbers
meas_dir = [0  0;0  0];                     % define measurement directions
invar = ssi_data3(predat.tdata',ii,refs);
[Aid,Cid,Qid,Rid,Sid] = ssi_data3(invar,n); % estimate state-space model
```

The prediction error, that is, the RMS error between the measured response and the response predicted by the identified model, can be computed as follows[1]:

```
Sigmaid = dlyap(Aid,Qid);                          % state correlation
Gid = Aid*Sigmaid*Cid' + Sid;                      % state-output correlation
Lambda0id = Cid*Sigmaid*Cid' + Rid;                % zero-lag correlation
Pid = dare(Aid',Cid',Qid,Rid,Sid,eye(size(A,1))); % Solve Riccati eq.
Kid = (Aid*Pid*Cid'+Sid)/(Rid+Cid*Pid*Cid');       % Kalman filter
%
ypred = 0*y;
z = zeros(size(Aid));
for par = 1:N,  % simulate response with identified forward innovation model
    z(:,par+1) = (Aid-Kid*Cid)*z(:,par) + Kid*y(:,par);
    ypred(:,par) = Cid*z(:,par);
end
%
prerr = zeros(2,1);
for par = 1:2   % compute prediction errors
    prerr(par) = sqrt(sum((y(par,:)-ypred(par,:)).^2)/sum((y(par,:)).^2))*100;
end
```

The computed prediction error is around 42 % for both channels. Although this might be surprising, it can be verified that this prediction error can not be reduced noticeably by increasing the data length. The Kalman filter yields optimal linear one-step ahead predictions [12], and 40 % is the best this filter performs for this system. As can be expected, much lower prediction errors have been reported for more lightly damped systems [7, sec. 8.3].

The time-domain decomposition of the measured response can be computed from only 3 MACEC functions. The following Matlab code illustrates this, and shows at ones how the decomposed signal can be plotted as in fig. 6.3.

[1] Note that for solving the Riccati equation, use is made of the Control Systems Toolbox® of MATLAB.

```
[fud,xi,Phi,quants,Ld,Km,Gm,refquants] = propmodpar5(Aid,[],Cid,[],Qid,...
    Rid,Sid,[],[],Ts,types,[],[],refs); % compute the modal parameters
[Lambda,Bm,Phi,D,Km] = modal2ss(fud,xi,Phi,Ld,Km,Ts,types);
ym = mdtime(y,Lambda,Phi,Km);
newfig([10 10])
h1 = subplot(4,1,1); plot(y(1,:)); set(gca,'xlim',[0 500],'ylim',5*[-1 1]);
h2 = subplot(4,1,2); plot(squeeze(ym(1,1,:))); set(gca,'xlim',[0 500],'ylim',5*[-1 1]);
h3 = subplot(4,1,3); plot(squeeze(ym(1,2,:))); set(gca,'xlim',[0 500],'ylim',5*[-1 1]);
h4 = subplot(4,1,4); plot(y(1,:).'-squeeze(ym(1,1,:)+ym(1,2,:)));
set(gca,'xlim',[0 500],'ylim',5*[-1 1]);
xlabel(h4,'time [s]'); ylabel(h1,'signal [m/s^2]'); ylabel(h2,'mode 1 [m/s^2]');
ylabel(h3,'mode 2 [m/s^2]'); ylabel(h4,'error [m/s^2]');
```

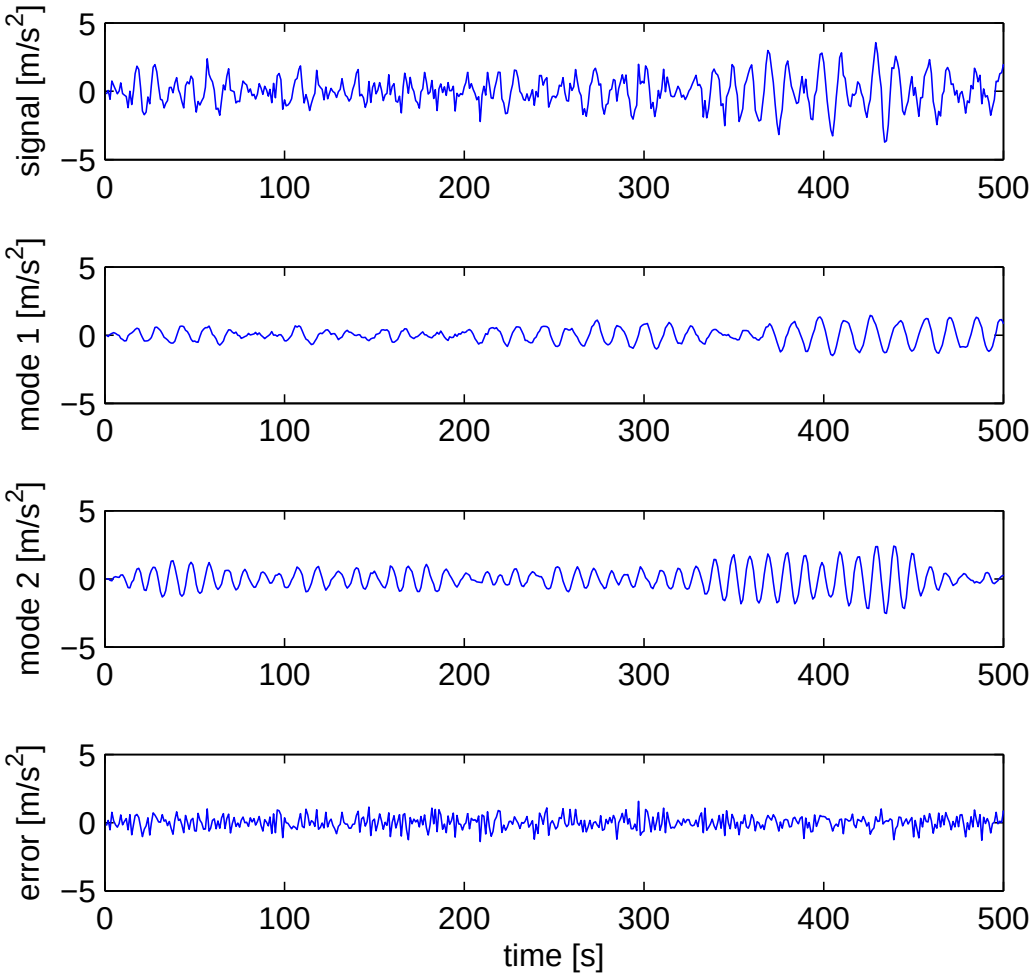


Figure 6.3: Verification example 1c: modal time-domain response decomposition from an output-only state-space model identified with SSI-data.

References

[1] H. Akaike. Stochastic theory of minimal realization. *IEEE Transactions on Automatic Control*, 19(6):667–674, 1974.

[2] B. Cauberghe. *Applied frequency-domain system identification in the field of experimental and operational modal analysis*. PhD thesis, Vrije Universiteit Brussel, 2004.

[3] M. Döhler and L. Mevel. Efficient multi-order uncertainty computation for stochastic subspace identification. *Mechanical Systems and Signal Processing*, 38(2):346–366, 2013.

[4] P. Guillaume, T. De Troyer, C. Devriendt, and G. De Sitter. OMAX - a combined experimental-operational modal analysis approach. In P. Sas and M. De Munck, editors, *Proceedings of ISMA2006 International Conference on Noise and Vibration Engineering*, pages 2985–2996, Leuven, Belgium, September 2006.

[5] W. Heylen, S. Lammens, and P. Sas. *Modal analysis theory and testing*. Department of Mechanical Engineering, Katholieke Universiteit Leuven, Leuven, Belgium, 1997.

[6] B. Peeters. *System identification and damage detection in civil engineering*. PhD thesis, Department of Civil Engineering, KU Leuven, 2000.

[7] B. Peeters and G. De Roeck. Reference-based stochastic subspace identification for output-only modal analysis. *Mechanical Systems and Signal Processing*, 13(6):855–878, 1999.

[8] B. Peeters and G. De Roeck. Stochastic system identification for operational modal analysis: A review. *ASME Journal of Dynamic Systems, Measurement, and Control*, 123(4):659–667, 2001.

[9] R. Pintelon and J. Schoukens. *System Identification*. IEEE Press, New York, NY, 2001.

[10] E. Reynders. *System identification and modal analysis in structural mechanics*. PhD thesis, Department of Civil Engineering, KU Leuven, 2009.

[11] E. Reynders. System identification methods for (operational) modal analysis: review and comparison. *Archives of Computational Methods in Engineering*, 19(1):51–124, 2012.

[12] E. Reynders and G. De Roeck. Reference-based combined deterministic-stochastic subspace identification for experimental and operational modal analysis. *Mechanical Systems and Signal Processing*, 22(3):617–637, 2008.

[13] E. Reynders, D. Degrauwe, G. De Roeck, F. Magalhães, and E. Caetano. Combined experimental-operational modal testing of footbridges. *ASCE Journal of Engineering Mechanics*, 136(6):687–696, 2010.

[14] E. Reynders, R. Pintelon, and G. De Roeck. Uncertainty bounds on modal parameters obtained from Stochastic Subspace Identification. *Mechanical Systems and Signal Processing*, 22(4):948–969, 2008.

[15] M. Schevenels. SIGFUN: a MATLAB toolbox for signal processing in civil engineering. Technical Report BWM-2006-02, Department of Civil Engineering, KU Leuven, February 2006.

[16] C.Y. Shih, Y.G. Tsuei, R.J. Allemang, and D. Brown. Complex mode indicator function and its applications to spatial domain parameter estimation. *Mechanical Systems and Signal Processing*, 2(4):367–377, 1988.

[17] P. Van Overschee and B. De Moor. *Subspace identification for linear systems*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.